Adaptive Techniques for Scale-Resolving Turbulence Simulations Using Super-Resolution Reconstruction

by

Miles J. McGruder

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in The University of Michigan
2024

Doctoral Committee:

Professor Krzysztof J. Fidkowski, Chair
Dr. Gary Coleman, NASA Langley Research Center
Professor Karthik Duraisamy
Professor Eric Johnsen

Miles J. McGruder

mmcgrude@umich.edu

ORCID iD: 0009-0009-4097-8138

Acknowledgments

I would like to start by thanking my advisor, Professor Krzysztof Fidkowski. He has has been a huge influence in my life. He is the one who got me started in fluid dynamics during his fall 2017 aerodyanmics class. It was really at that point that college got interesting for me, and it is safe to say I have stayed a bit longer than I expected. I would also like to thank Professor Duraisamy, Professor Johnsen, and Dr. Coleman for serving on my committee. It is only with their constructive feedback that I have been able to finish this thesis.

Countless students have made FXB home over the years. My old lab mates Devina, Yifan, Gary, Gustavo, and Vivek really got me started in the department before the pandemic. Nick, Fabian, and Logan went out of their way to connect with me as a new student. Studying for prelims with Alex and Rahul really kept me going while sitting at my parents' dining room table. Playing squash with Hoang and Guodong was my first real exercise in years. Thanks again to Nick for hosting game nights and F1 watch parties where I met Chris, Jasmin, and other members of the CASLAB. Thanks to Rakesh and Marco for being great roommates. Thanks to Alex for being a great office mate over the years, it is always fun to have someone around as excited about CFD as I am. I have loved talking to our office neighbors Sebastian, Elliot, and Ral about numerical methods. The MDO lab has shaped my post-pandemic experience in the most positive way. Huge thanks to Eytan, Hannah, and Alex for biking with me so often. Thanks to Sabet and Kleb for teaching me to solve nonlinear systems. Thanks to Ella, Ali, and Feriel for all the insightful discussions over the years. Of course, a huge thank you to Bernardo for keeping me gonig while writing this thesis. Finally, thank

you to my family for supporting me every step of the way. I couldn't have done it without all of you.

Thank you to NASA for funding the work in this thesis under the support of NASA Cooperative Agreement 80NSSC18M0149.

Table of Contents

List of Figures

List of Tables

Abstract

Accurate prediction of turbulent flow phenomena is an area of keen engineering interest. Predicting these phenomena, such as flow separation, remains difficult after decades of research. The Reynolds averaged Navier-Stokes (RANS) equations are commonly used, but rely on empirical models that can fail to accurately predict interesting phenomena like separation. Direct numerical simulation (DNS) would solve all the shortcomings of the RANS equations, but it is not practical on modern machines at Reynolds numbers of engineering interest. Large-eddy simulation (LES) is a crucial middle ground that is becoming increasingly useful as computational power grows. LES relies on empirical models for small-scale flow features that are computationally expensive to capture, but still resolves larger turbulent features. While small-scale modeling brings practical turbulence simulations just within reach of modern machines, these simulations remain expensive. Adaptation can increase the practicality of LES by further reducing its computational cost.

This dissertation implements an adaptive method for LES in a discontinuous Galerkin (DG) context. The core of the adaptation process is a neural-network that predicts fine scales in a given flow-field. Preliminary testing in 1D shows reconstruction is accurate with a simple neural-network. Reconstruction remains accurate in two and three dimensions using simple network architectures. Preliminary testing on more sophisticated network architectures indicates significant gains in reconstruction accuracy are possible. A single super-resolution network trained on a variety of data is used to reconstruct various flows during adaptation. Two error indicators are proposed based on super-resolution reconstruction. One is based on the magnitude of the network's fine scale correction and the other on an entropy-adjoint

weighted residual. The error indicators are tested by adapting a variety of turbulent flow problems.

The adaptive method outperforms uniform refinement given a poor initial mesh. The adaptive methods are tested on a channel flow problem where no initial mesh refinement is assumed. The simple correction magnitude error indicator returns the expected error pattern across Reynolds numbers. Adaptation with this indicator proceeds as expected, placing the most resolution near the channel walls, while leaving the channel center unrefined. The entropy-adjoint weighted residual indicator shows more noise using the same averaging process. The extra noise decreases as Reynolds number increases. Adaptation with the correction magnitude indicator generally outperforms uniform refinement on this case, achieving similar performance to uniform refinement at approximately 25% fewer degrees of freedom. The correction magnitude indicator is then tested against a mean velocity gradient indicator. Testing on a periodic hill geometry shows similar results between the error indicators. The indicators show high error near the center of the domain, as opposed to the already refined domain edges. Adaptation performance is more similar to uniform refinement on this case. Finally, a modified version of the simple correction magnitude indicator is tested on a geometry intended to mimic a trailing edge cooling slot in turbomachinery. A tendency toward wake refinement continues from the periodic hill test case. Once again, the adapted results are in line with, or slightly better than uniform refinement on a degree of freedom basis. Suggestions are made to improve the performance of the weighted residual error indicator with different averaging techniques. Suggestions are also made for impactful alternative research directions.

Chapter 1

Introduction

Today, the engineering design process is governed primarily by low-fidelity models. A deep knowledge of physical principles informing low-fidelity models and a thoughtful engineer can work wonders. Rather than replace these processes, computational fluid dynamics (CFD) seeks to provide precise answers to questions that cannot be answered by simple models. These more difficult questions often involve viscous phenomena, such as flow separation and drag prediction. Developing accurate models of viscous phenomena has been an active research area for decades. The difficulty stems from the complexity of solutions to the Navier-Stokes equations.

The Navier-Stokes equations have proved to be an extremely accurate macroscopic model of fluid flow. Accuate solutions to these equations would answer most analysis questions of engineering interest. Unfortunately, these equations admit turbulent solutions at sufficiently high Reynolds numbers. These solutions exhibit a wide range of spatial and temporal scales, making the equations extremely difficult to solve.

In a world of infinite computing capacity, direct numerical simulation (DNS) would be the ubiquitous solution. DNS simply resolves all spatial and temporal scales of the solution. The only requirement is a computational grid able to resolve the Kolmogorov microscales and a

time step to match. In this reality, engineers could move high-fidelity analysis forward in the design process. Costly design mistakes due to the inevitable inaccuracies of low-fidelity models would be avoided. Unfortunately, despite rapid advances in parallel computing over the last decade, DNS remains out of reach for all but moderate Reynolds numbers and the simplest goemetries.

The Reynolds-averaged Navier-Stokes equations have become common in practical engineering applications. These equations rely on the Reynolds decomposition, where flow-field variables are separated into time-averaged and fluctuating components. The relevant equations are therefore a time-averaged version of the Navier-Stokes equations. Thankfully, most phenomena of engineering interest are statistically stationary so a time-averaged set of equations may be applied. These equations would be perfect were it not for the nonlinear Reynolds stress term in the equations, which must be modeled empirically.

To increase accuracy, a higher-fidelity method is required that remains short of DNS. Large-eddy simulation (LES) fits this role nicely.

1.1 Large-Eddy Simulation

LES [83, 62, 102, 82] aims to resolve scales of interst, that is, those larger than the computational grid resolution. Smaller scales are handled by subgrid-scale models that mimic the effects of fine scale turbulence on the larger scales. Instead of averaging field quantities over time, LES typically applies a low-pass filter to scales smaller than the grid resolution itself. These filtered equations then require a subgrid-scale model to close.

LES accepts subgrid-scale modeling errors for the sake of vastly reduced computational cost relative to DNS. Turbulence is a chaotic phenomenon. Any solution error will grow exponentially as it propagates through time. Eventually, the resulting solution will bear no resemblance to the initial condition. This may make LES sound useless at first, except

we are interested in time-averaged flow properties, not the exact time evolution. LES is perfectly capable of producing nearly correct time-averaged fields, even if the instantaneous eddy positions are incorrect.

LES is typically performed with an explicit subgrid-scale model, for example the Smagorinsky model [108]. An explicit subgrid-scale model is not required to perform LES. Plenty of examples exist of LES performed without subgrid-scale models in finite-volume [59] and finite-element [115, 28, 91, 90, 11, 9, 53] settings. These approaches are called implicit large-eddy simulation (ILES). In this case, the dissipation of the method itself acts as the subgrid-scale model. One could consider these techniques under-resolved DNS because the only difference between running ILES and DNS is grid resolution. Increasing the grid resolution of on ILES until it can resolve the Kolmogorov microscales results in a DNS solution.

1.2 Error Estimation

Despite advances in computational power and modeling small flow-field scales, LES remains too computationally expensive for most practical situations. We could further reduce costs by concentrating mesh resolution only where necesary in the computational domain. This process can be performed manually, but takes additional time, effort, and can still result in a sub-optimal grid. It would be much easier and more effective to automatically concentrate resources where required in the domain. For this, we need a form of error-estimation for chaotic turbulent flows.

Adjoint-based error-estimation [45, 40] has proved worthwhile on steady and some unsteady flow problems. The adjoint method is output-based, meaning that error in scalar quantities of interest computable from the flow-field variables, such as drag, is estimated. The technique can do so because the adjoint itself is a linear sensitivity measurement between the output of interest and the residual. This error estimate is localizable since the error estimate for the full domain reduces to a sum over elements. Taking the element contributions separately

and, commonly, taking the absolute value of each one yields a local error indicator.

Ideally we would compute error relative to the exact differential equation solution. Of course, obtaining the exact solution would be prohibitively expensive and defeat the entire purpose of the error-estimation exercise. When computing an adjoint-based indicator we instead settle for two finite-dimensional discretization levels. Computing the error estimate requires an adjoint approximation in the finer discretization level. Solving the adjoint equation in the fine space could be computationally expensive. Solving for the adjoint requires the solution of a linear-system whose size is the same as the system of discretized equations we are trying to solve. Approximate solutions are appropriate in this context, since the error estimate is not an error bound.

While the adjoint can be used in some laminar unsteady settings, chaotic adjoints have fundamental difficulties. The adjoint is a linear relationship between residuals and outputs. The fundamental characteristic of a chaotic system is that the future state of the system is extremely sensitive to initial conditions. This means that any adjoint sensitivity will become meaningless over a relatively short time span. This causes solutions to the adjoint equation to diverge, or at least return inaccurate results. While the adjoint equation alone cannot serve our purpose, an approximate reconstruction route remains available.

An alternative to the traditional output adjoint is the entropy-adjoint proposed by Fidkowski and Roe [47]. They showed that the entropy variables are an adjoint for a particular output functional related to spurious entropy generation in the computational domain. The spurious entropy generation is closely related to under-resolution in the domain. While the entropy output cannot target outputs as precisely as an output adjoint, adaptation with the entropy-adjoint correlates well with outputs of interest [43]. The entropy-adjoint approach extends to unsteady simulations [71]. Crucially, the entropy-adjoint is a route to adaptation for chaotic fields [10]. Since no solution of the adjoint equations is required, only a variable transformation, the entropy-adjoint is readily accessible.

Alternative appraches for chaotic sensitivity analysis have been developed. The ensemble adjoint method [78, 37] involves averaging sensitivities computed over many system realizations. In practice, the method converges too slowly to be practical. Another approach is the least-squares-shadowing technique proposed by Wang *et al.* [117]. Finding the shadow trajectory requires solving a, generally large, constrained optimization problem [15, 14]. On problems of practical size, the method computes accurate sensitivities but computational cost is high [16]. The non-intrusive least-squares-shadowing approach has been developed by Ni and Wang [94] to address the cost issues of the previous method. Testing in a discrete adjoint setting by Blonigan [13] shows that the method remains expensive. An alternative cost reduction approach based on reduced order modeling has been proposed by Shimizu and Fidkowski [104, 105].

Other simple strategies for adaptive sensors in chaotic flows include an indicator based on the ratio of resolved to total kinetic energy proposed in [24]. Kinetic energy present at fine scales can also be used to generate an anisotropic error indicator [112]. Discretization and modeling error in LES have been considered separately in a finite element framework by Hoffman in [64]. Previous work by Bassi *et al.* [8] has considered a combined error indicator based on pressure jumps [76], and modal coefficient decay [98]. Simple gradient-based indicators [6, 60, 97] have been shown to miss key flow features [118]. Venditti and Darmofal [116] showed that output-based indicators ensure key flow features are captured and therefore outperform gradient-based indicators.

1.3 Mesh Adaptation

Localized error indicators are typically formed element-wise. The element local error indicator says nothing about the way the mesh should be adapted, leading to a variety of techniques. One option is hanging-node adaptation, used in [12, 70, 100, 30, 106, 25]. In a typical computational mesh, each face on each element meets a single face on an adjacent

element. This restriction is not strictly necessary, since the integration over a face that informs the residual contribution in finite-volume and discontinuous finite-element methods can be broken up. An element face could then abut faces of two different elements, as long as integration is handled accordingly. This is the idea behind hanging-node meshes. Now, given an element-local error estimate, a high-error element may simply be split into several smaller elements without remeshing the nearby region.

In a discontinuous finite-element context, a local error indicator can be used to alter the element's polynomial approximation order. This order, or "p", adaptation has been used in several previous works [10, 114, 92]. The method is simple to implement. The most complex requirement is a finite-element code that supports variable polynomial order. This is mostly a matter of having sufficiently sophisticated book-keeping structures. One must also ensure that integration on element faces is handled correctly on faces where adjacent elements have different orders. Like the hanging-node case, the independence of the elements makes this relatively straightforward.

For more precise control over the mesh, some sort of remeshing should be employed. Smooth remeshing avoids abrupt resolution changes in the computational domain. Remeshing also facilitates element-shape optimization. Yano and Darmofal [123, 124] developed a framework to produce an optimal mesh for a given number of degrees of freedom. This framework allows anisotropic-mesh optimization using only element-local error indicators.

Mesh adaptation is possible not only in space, but also time [46, 41]. Even anisotropic-mesh generation in higher dimensions is possible. For example, Caplan *et al.* [22, 21] perform anisotropic-mesh optimization in four dimensions. This allows optimal mesh construction not only through space, but also time. This work requires a solver able to handle space and time discretization simultaneously. Fortunately, the discontinuous Galerkin finite-element method can be extended to cover time and space simultaneously, providing an appropriate setting for these optimizations.

In adaptive LES, Ims and Wang [68] recently reviewed several efforts. The first method is based on an unsteady residual indicator originally used in [51]. The second indicator is the spectral decay indicator mentioned earlier [98], and also used by Bassi *et al.* in [8]. The third indicator, developed by Toosi and Larsson [112], is an anisotropic indicator based on the energy in the smallest resolved scales. They use this indicator to predict optimal grid anisotropy. Toosi and Larsson extend their previous work with an error indicator that measures the dependence of the solution on the LES filter width [113]. Bassi *et al.* [10] have performed LES adaptation with an entropy-adjoint-weighted-residual error indicator in a p-adaptive setting. Abbà *et al.* [1] also use a p-adaptive framework for their high-order LES adaptation.

1.4 Machine Learning for Adaptation

Machine-learning techniques based on artificial neural networks have become increasingly common over the past decade. The dual tail winds of increasing computational power and widely available data have led to the widespread usefulness of these machine-learning techniques. The mid 2000s saw an explosion in parallel computing as CPU clock speed scaling began to slow. Researchers began using graphics processing units (GPU) for more general and highly parallel computing tasks. Early GPU work by machine-learning researchers, [77] for example, have lead to the explosion we see today.

This proliferation has extended to CFD, and mesh adaptation in particular. For example, Fidkowski and Chen [44] present a machine-learning technique to determine adapted mesh anisotropy. The method compares well against the framework introduced by Yano mentioned earlier. Chen and Fidkowski [26] also propose a convolutional neural network for error-estimation. The goal is to use the network as a surrogate model for the error estimate and adaptive flagging based on readily available flow features. In a similar vein, Bohn and Feischl [17] show that a recurrent neural network can be trained to optimally estimate error and flag

elements for a variety of differential equations. Some other recent works learn an adaptation policy directly using a reinforcement learning framework [121, 48]. Fidkowski [42] uses field inversion and machine learning (FIML) [95, 107, 65, 63] to perform output-based adaptation of chaotic flow. FIML is used to correct a RANS model to match time averaged unsteady data. Output-adjoint-based adaptation methods can then be applied on the corrected RANS equations. Tlales *et al.* [111] use machine learning to differentiate laminar from turbulent flow regions in a technique that does not require tunable parameters.

Another way to use machine learning for adaptation is to perform reconstruction. In recent years, machine learning has become the dominant tool for super-resolution [122, 4]. Super-resolution is a classical computer vision task where a low-quality image is restored to a high-quality baseline using an upscaling model [39, 96]. In a fluid dynamics context, Liu *et al.* [84] proposed an approach for spatio-temporal super-resolution using multiple convolutional paths, each handling different time ranges. Fukami *et al.* [50] proposed an alternative temporal super-resolution architecture built on their previous single-image work [49]. The generative adversarial approach of Deng *et al.* [29] successfully resolves large wake fields with 4x and 8x upscaling factors. In finite elements, Pradhan and Duraisamy [99] have introduced a variational-multiscale-consistent network architecture for the discovery of Galerkin discretization closures. We can take inspiration from their work to develop models able to perform adaptation on turbulent flow problems.

1.5 Objective and Contributions

An output-based error estimation technique, like the adjoint-weighted-residual, would be ideal when adapting turbulent flows. However, we know that computing an output adjoint is reliant on sensitivity analysis that fails in a turbulent setting. An entropy-adjoint-weighted-residual approach could be used instead, so that only a variable transformation is required [47]. A fine-space adjoint is still required for an error indicator, it can be found by recon-

struction from neighboring states [45]. This approach was taken by Bassi *et al.* in [10], using a simple interpolation procedure on the time-averaged state of neighboring elements for the adjoint reconstruction. The reconstruction procedure does not explicitly attempt to resolve fine-scale flow features. To the knowledge of the author, reconstruction techniques that generate fine-scale turbulent features have not been used to create error indicators for turbulent adaptation. Our objective is to introduce such a fine-scale reconstruction technique to the adaptation of turbulent flows. We will select an appropriate reconstruction model, generate error indicators based on that model, and perform adaptation on various turbulent problems.

The main contributions of this dissertation are:

1. **Introduced error indicators based on super-resolution reconstruction.** One indicator is based on the magnitude of the correction requested by the super-resolution reconstruction model. The other indicator uses the reconstruction model in an entropy-adjoint-weighted-residual framework.

2. **Demonstrated that a single reconstruction model trained on a variety of flows is sufficient for adaptation.** In one dimension, we show that reconstruction is nearly exact on unseen test snapshots when training and test snapshots are of the same parameterization. In two dimensions, reconstruction quality degrades slightly and modifications to the network architecture are introduced to further improve reconstruction quality. In three dimensions, a reconstruction model trained at a variety of flow Reynolds numbers maintains high reconstruction quality relative to more specialized counterparts.

3. **Implemented reconstruction-based adaptation in a high-order discontinuous Galerkin code.** We use NASA's *eddy* for all 3D adaptations. Super-resolution training, reconstruction, and error indicators, have been added to *eddy*.

4. **Performed super-resolution-based adaptation on several chaotic flow problems.** The first problem is a turbulent channel where no initial grid refinement has

been assumed. Error indicators show high error near channel walls in the region of high turbulent kinetic energy. The next problem is a periodic hill, a canonical case of flow separation. Adaptation focuses on the separation wake downstream of the hill. Finally, a geometry designed to mimic a cooling slot in turbomachinery is tested. Adaptation concentrates in the wake region behind a bluff body above the cooling slot.

1.6 Outline

In Chapter 2 we briefly discuss the discontinous Galerkin finite-element method. Two equation sets are used in this dissertation. The Kuramoto-Sivashinsky (KS) equation is used for preliminary reconstruction testing while the Navier-Stokes equations are used for final turbulent adaptive testing. Both are discussed with an emphasis on the Kuramoto-Sivashinsky equation because it has relatively little discussion elsewhere.

Chapter 3 deals with reconstruction testing in one dimension. It introduces super-resolution reconstruction, a canonical computer vision problem that will be repurposed in this dissertation for the reconstruction of finite-element states. A high-order KS-equation implementation is verified. High-order data is required since network training and testing is projection based. Basic reconstructin is tested establishing the plausibility of super-resolution reconstruction as the core of an error indicator. Reconstruction is shown to generalize well to unseen data for the same equation parameterization and mesh. Moving across parameterizations is shown to be more challenging, but training on a variety of data largely alleviates this issue.

Chapter 4 moves into higher dimensions. This chapter shows that the reconstruction quality from 1D testing does deteriorate in higher dimensions. However, the reconstruction is always directionally correct, as measured by energy spectra. It is also shown that reconstruction quality can be significantly improved with simple changes to network architecture. Leaving that aside and moving forward with a simpler architecture, we determine appropriate network size and training set information. A single network trained on a variety of data is chosen for

adaptive use.

Chapter 5 finishes the results by applying the previously gained reconstruction knowledge to adaptation. Two error indicators are introduced. One measures the state correction predicted by the network, while the other attempts an entropy-adjoint-weighted-residual approach. Spatial and temporal averaging techniques are discussed. Adaptation is then sanity tested on a turbulent channel case with uniformly distributed elements at a variety of friction Reynolds numbers for both error indicators. This adaptation is then tested on a periodic hill geometry at low Reynolds number. Finally, a large-scale test case with significant wake flow is included.

Chapter 2

Discretization Techniques

2.1 Introduction

In this chapter we discuss the discretizations and solvers used in this work. The Navier-Stokes equations are introduced in compact form. We use *eddy*, a high-order DG code from NASA Ames Research Center, to solve the Navier-Stokes equations. Discussion is included of *eddy*'s many useful capabilities. We then move on and discretize the 1D Kuramoto-Sivashinsky equation. This equation is a canonical case of turbulence in 1D. The fourth order term makes its DG discretization challenging.

2.2 Discontinuous Galerkin Finite-Element Method

This work uses the discontinuous Galerkin finite-element method throughout. Two equation sets are discretized and used: the Navier-Stokes equations and the Kuramoto-Sivashinsky equation. We will begin by discussing Navier-Stokes, then move on to Kuramoto-Sivashinsky.

Parts of this chapter appear in or are adapted from our previously published papers [86, 85].

In compact form the Navier-Stokes equations are

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \vec{\mathbf{F}}(\mathbf{u}) - \nabla \cdot \vec{\mathbf{G}}(\mathbf{u}, \nabla \mathbf{u}) = \mathbf{0}, \tag{2.1}$$

where $\mathbf{u} \in \mathbb{R}^s$ is the rank s state vector, $\vec{\mathbf{F}}$ is the inviscid flux, and $\vec{\mathbf{G}}$ is the viscous flux.

DG splits a computational domain $\mathbf{\Omega}$ into a tessellation $\mathcal{T}_h$ of non-overlapping elements, each covering a volume $\mathbf{\Omega}_e$. The state is represented as a linear combination of basis functions

$$\mathbf{u} = \sum_{e}^{N_e} \sum_{i}^{N_b} \mathbf{U}_{e,i} \boldsymbol{\phi}_{e,i}, \tag{2.2}$$

where N_e is the number of elements, N_b is the number of basis functions on element e, $\mathbf{U}_{e,i}$ is basis function coefficient i on element e and $\boldsymbol{\phi}_{e,i}$ is the i^{th} basis function on element e. Each set of basis functions has support over only a single element. We can formally consider each state $\mathbf{u}_h$ to be a member of a solution-approximation space $\boldsymbol{\mathcal{V}}_h = [\mathcal{V}_h]^s$ with $\mathcal{V}_h$ defined as

$$\mathcal{V}_h = \{u \in L_2(\mathbf{\Omega}) : u|_{\mathbf{\Omega}_e} \in \mathcal{P}^{p_e} \; \forall \mathbf{\Omega}_e \in \mathcal{T}_h\}, \tag{2.3}$$

where $\mathcal{P}^{p_e}$ is the set of polynomials of order p_e on element e. To discretize and solve the system of equations, we first find the weak form of the Navier-Stokes equations. To do so we multiply Equation 2.1 by test functions, integrate by parts, and couple elements via numerical fluxes:

$$\begin{aligned} \int_{\mathbf{\Omega}_e} \mathbf{w}_h^T \frac{\partial \mathbf{u}}{\partial t} \, \mathrm{d}\mathbf{\Omega} - \int_{\mathbf{\Omega}_e} \nabla \mathbf{w}_h^T \cdot \left[\vec{\mathbf{F}}(\mathbf{u}_h) - \vec{\mathbf{G}}(\mathbf{u}_h, \nabla \mathbf{u}_h) \right] \mathrm{d}\mathbf{\Omega} + \\ \int_{\partial \mathbf{\Omega}_e} \mathbf{w}_h^T \left[\widehat{\mathbf{F}}\left(\mathbf{u}_h^+, \mathbf{u}_h^-\right) - \widehat{\mathbf{G}}\left(\mathbf{u}_h^+, \mathbf{u}_h^-, \nabla \mathbf{u}_h^+, \nabla \mathbf{u}_h^-\right) \right] \cdot \vec{n} \, \mathrm{d}S - \\ \int_{\partial \mathbf{\Omega}_e} \left(\mathbf{u}_h^+ - \{\mathbf{u}_h\}\right)^T \vec{\mathbf{G}}\left(\mathbf{u}_h^+, \nabla \mathbf{w}_h^+\right) \cdot \vec{n} \, \mathrm{d}S = \mathbf{0}, \; \forall \mathbf{w}_h \in \boldsymbol{\mathcal{V}}_h. \end{aligned} \tag{2.4}$$

the quantity $\partial \mathbf{\Omega}_e$ represents the element boundary, and on that boundary, $(\cdot)^+$ and $(\cdot)^-$ represent quantities taken from the current and neighboring element, respectively. Approximate

numerical fluxes are denoted by $\widehat{(\cdot)}$, $\{\cdot\}$ represents a face average or boundary value, and $\vec{n}$ is the outward pointing normal vector. Time evolution consists of solving the unsteady residual equation $\mathbf{R}'_h(\mathbf{U}_h)$ at each time step. The equation is

$$\underbrace{\mathbf{R}'_h(\mathbf{U}_h)}_{\text{unsteady residual}} \equiv \underbrace{\mathbf{M}_h}_{\text{mass matrix}} \frac{d\mathbf{U}_h}{dt} + \underbrace{\mathbf{R}_h(\mathbf{U}_h)}_{\text{steady residual}} = \mathbf{0}, \tag{2.5}$$

where $\mathbf{M}_h$ is a mass matrix resulting from the combination of basis and test functions from the unsteady time term.

2.2.1 *eddy*

To create our Navier-Stokes training and testing data we use *eddy*, a DG solver from NASA Ames Research Center [31, 33]. *eddy* is designed to be efficient at very high polynomial orders, for example $p = 15$. It uses a matrix free Newton-Krylov solver to alleviate Jacobian storage requirements on large systems. The Newton-Krylov solver is combined with an alternating-direction-implicit (ADI) preconditioner developed for space-time tensor product elements [34]. These features make the solver efficient for large-scale, high-order problems. In addition to supporting high-order elements, *eddy* also supports variable order elements. This is crucial, since we intend to perform order adaptation with our newly developed error indicators.

When performing order adaptation, the initial mesh will consist of large elements resulting in a poorly resolved flow-field. Sufficient under-resolution could destabilize the solver. Thankfully, *eddy* is designed around an entropy variable formulation [33], ensuring nonlinear stability. Some test cases in Chapter 5 begin severely under-resolved. We have observed that other solvers cannot successfully evolve the initial under-resolved flow-fields. We also use the multi-physics capabilities of *eddy* for turbulent inflow generation [23], and perfectly matched layer (PML) far field conditions [52].

Modern CPUs support vector instructions, allowing many, *e.g.* 8, floating point operations to be performed in a single instruction. Efficient use of vector instructions is required to reach the full compute capacity of any modern CPU. *eddy* is designed with support for vector instructions in mind. It uses a combination of optimized linear algebra routines and compiler vectorization to achieve high-performance with vector instructions. As a result, *eddy* is most efficient when $p + 1$ is a power of two. The optimization for specific orders can turn into a down side for p-adaptation, our version of *eddy* only supports odd p.

2.3 Kuramoto-Sivashinsky (KS) Equation

We intend to test the fundamentals of super-resolution adaptation in a p-adaptive finite-element framework. To this end, we need to choose a discontinuous Galerkin finite-element discretization of the KS equation. We discretize the KS equation in the form

$$u_t + uu_x + u_{xx} + \nu u_{xxxx} = 0. \tag{2.6}$$

Only the viscosity coefficient, ν, on the fourth-order diffusion term is varied. Increasing or decreasing ν is sufficient to explore the range of possible solutions, from steady-state to chaos. Boundary conditions are always periodic except in particular cases which we will discuss later. We proceed by discussing the discretization of each term individually, beginning with the spatial terms.

2.3.1 Nonlinear Term

The nonlinear term, uu_x, promotes mixing of scales as the solution evolves in time. This can be intuited from the fact that this term is identical to the nonlinear term in Burger's equation. In the exact solution to Burger's equation, wave speed is identical to the state at any given place and time. This causes flow features to "collapse" into each other creating shocks. In our case, the "collapse" only causes mixing since diffusion prevents sharp discontinuities.

For the discontinuous Galerkin discretization of first-order terms, we need to multiply by test functions and integrate over the domain. Note that test and basis functions only have support over a single element, so this integration reduces to integration over individual elements

$$\int_{\Omega_k} v_{k,i} \left(\frac{\partial u}{\partial t} + \frac{\partial f}{\partial x} \right) \mathrm{d}x = 0 \tag{2.7}$$

where $\boldsymbol{\Omega}_k$ is the interior of element k, $v_{k,i}$ is the i^{th} test function on element k, u is a (scalar) state, and $f = \frac{1}{2}u^2$ is the flux. After an intermediate integration by parts we get

$$\int_{\Omega_k} v_{k,i} \frac{\partial u}{\partial t} \, \mathrm{d}x - \int_{\Omega_k} \frac{\partial v_{k,i}}{\partial x} f \, \mathrm{d}x + \left[v_{k,i} \hat{f} \right]_{x_{k-1/2}}^{x_{k+1/2}} = 0, \tag{2.8}$$

where $\hat{f}$ is an approximate numerical flux to be defined below, and $k + 1/2$ and $k - 1/2$ are the right and left sides of the element, respectively.

Knowing the above DG discretization, we need to find appropriate fluxes for the nonlinear term. First we will convert the term to flux form via the simple transformation

$$u u_x = \left(\frac{1}{2} u^2 \right)_x \implies f = \frac{1}{2} u^2. \tag{2.9}$$

With our analytical flux in hand, we can handle inter-element discontinuities with an approximate numerical flux. A nonlinear upwind flux is selected with the form

$$\hat{F} = \frac{1}{2} \left(f_j \big|_{j+1/2} + f_{j+1} \big|_{j+1/2} \right) - \frac{1}{2} \left| \hat{a}_{j+1/2} \right| \left(u_{j+1} \big|_{j+1/2} - u_j \big|_{j+1/2} \right), \tag{2.10}$$

where $(\cdot)|_{j+1/2}$ denotes quantities evaluated at the interface between elements, $(\cdot)_j$ denotes a quantity from the left element, and $(\cdot)_{j+1}$ denotes a quantity from the right element. The choice of wave speed at the interface, $\hat{a}_{j+1/2}$, determines the numerical scheme. In this work

we have used

$$\hat{a}_{j+1/2} = \begin{cases} \left.\frac{f_{j+1}-f_j}{u_{j+1}-u_j}\right|_{j+1/2} & \text{when} \quad u_j\big|_{j+1/2} \neq u_{j+1}\big|_{j+1/2} \\ f\left(u_j\big|_{j+1/2}\right) & \text{when} \quad u_j\big|_{j+1/2} = u_{j+1}\big|_{j+1/2}\,. \end{cases} \tag{2.11}$$

The choice $\frac{f_{j+1}-f_j}{u_{j+1}-u_j}$ is consistent with the design of Roe schemes.

2.3.2 Anti-diffusive Term

The anti-diffusive term, u_{xx}, promotes growth in the solution. In the absence of the diffusive term u_{xxxx}, the solution would simply blow up. For this anti-diffusion term we use the classical second-order interior-penalty method [36].

Finite-element methods are often presented in a general bilinear form, involving state and test components, over a full computational domain. For the purpose of implementation, it is useful to restrict this general formulation to an element-specific residual calculation. In the nomenclature of Arnold *et al.* in [5], the former is the primal form and the latter is the flux form of the discretization.

We begin by defining the relevant trace operators, the average, $\{\cdot\}$, and jump $[\cdot]$ on the interface of elements 1 and 2. The average and jump are defined as

$$\{q\} = \frac{1}{2}\left(q_1 + q_2\right), \qquad [q] = q_1 n_1 + q_2 n_2, \tag{2.12}$$

where q is any given quantity, and n is the outward pointing normal from the indicated element. Literature will typically at this point define the same trace operators for vector quantities. In our case, we are restricting ourselves to the 1D KS equation, which is scalar, so the vector trace operators reduce to their scalar versions.

With the trace operators introduced, the bilinear form for the classical second-order interior-

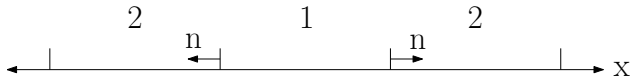


Figure 2.1: The primal form may be converted to a useful form for implementation by specifying one of the two elements in the jump terms as the element of interest.

penalty method is

$$B(u,v) = \int_\Omega u_x v_x \,\mathrm{d}x \; - \int_\Gamma [u]\{v_x\} + \{u_x\}[v] \,\mathrm{d}s \; + \int_\Gamma \frac{\eta}{h}[u][v] \,\mathrm{d}s, \tag{2.13}$$

where Γ is the set of all element interfaces in the domain, η is a positive number, and h is a measure of element size normal to the face. The final term is separated for emphasis; it is the jump penalization term responsible for keeping the method stable as long as η is set sufficiently high. In this work, $\eta = 8$ has been used, and h is set to the element length.

While the primal form is useful for proving convergence properties of the method, it does not provide a residual contribution corresponding to each test function on each element. Next we will lay out the necessary assumptions to convert the primal form into the flux form for implementation.

Conversion from the primal to flux form can be performed by following Figure 2.1. When we expand the primal form with the trace operators, we will assume that element 1 is our element of interest, and element 2 is any adjacent element. Further assuming the mesh is water tight, we know that $n_2 = -n_1$. We denote quantities from the element of interest with $(\cdot)^+$, and quantities from adjacent elements with $(\cdot)^-$. We restrict the computational domain Ω, to the domain of element k, Ω_k, and the set of edges Γ, to the element's edges Γ_k. Finally, we know that test functions have only local support over each element, so all test function values from neighboring element elements are simply zero. These operations result in the

discrete residual

$$R(k,i) \mathrel{+}= \int_{\Omega_k} \frac{\partial u}{\partial x}\frac{\partial v_{k,i}}{\partial x}\,\mathrm{d}x - \int_{\Gamma_k} \frac{v_{k,i}}{\partial x}\frac{1}{2}\left(u^+ - u^-\right) n\,\mathrm{d}s - \int_{\Gamma_k} v_{k,i}\left[\frac{1}{2}\left(\frac{\partial u^+}{\partial x} + \frac{\partial u^-}{\partial x}\right) - \frac{\eta}{h}\left(u^+ - u^-\right) n\right] n\,\mathrm{d}s, \tag{2.14}$$

for test function i on element k, where n is the outward facing normal, in our case simply -1 or 1. With these operations we have recovered the same form as the nonlinear term.

2.3.3 Fourth-Order Diffusion Term

Fourth-order interior-penalty discretizations have been discussed in several publications including [109, 56, 57]. Working from the bilinear forms derived in these papers, we can follow the same process as the second-order term to find the flux form for implementation. For this solver, we begin from the bilinear form listed in [57]

$$B(u,v) = \int_\Omega \frac{\partial^2 u}{\partial x^2}\frac{\partial^2 v}{\partial x^2}\,\mathrm{d}x + \int_\Gamma \left(\left\{\frac{\partial^3 u}{\partial x^3}\right\}[v] + \left\{\frac{\partial^3 v}{\partial x^3}\right\}[u] - \left\{\frac{\partial^2 u}{\partial x^2}\right\}\left[\frac{\partial v}{\partial x}\right] - \left\{\frac{\partial^2 v}{\partial x^2}\right\}\left[\frac{\partial u}{\partial x}\right] + \sigma[u][v] + \tau\left[\frac{\partial u}{\partial x}\right]\left[\frac{\partial v}{\partial x}\right]\right)\mathrm{d}s, \tag{2.15}$$

where there are two new stabilization constants σ and τ. They penalize the state and gradient across each interface respectively. These constants must be set sufficiently high to ensure the stability of the method. In this work σ is set to p^6 for $p \leq 3$ and 729 otherwise, while τ is set to p^2 for $p \leq 3$ and 9 otherwise. Proceeding with our substitution process, we

find the flux form appropriate for implementation:

$$R(k,i) \mathrel{+}= \int_{\Omega_k} \frac{\partial^2 u}{\partial x^2}\frac{\partial^2 v_{k,i}}{\partial x^2}\,\mathrm{d}x + \int_{\Gamma_k} \frac{\partial^3 v_{k,i}}{\partial x^3}\frac{1}{2}\left(u^+ - u^-\right)n + v_{k,i}\frac{1}{2}\left(\frac{\partial^3 u^+}{\partial x^3} + \frac{\partial^3 u^-}{\partial x^3}\right)n - \frac{\partial^2 v_{k,i}}{\partial x^2}\frac{1}{2}\left(\frac{\partial u^+}{\partial x} - \frac{\partial u^-}{\partial x}\right)n - \frac{\partial v_{k,i}}{\partial x}\frac{1}{2}\left(\frac{\partial^2 u^+}{\partial x^2} + \frac{\partial^2 u^-}{\partial x^2}\right)n + \sigma v_{k,i}\left(u^+ - u^-\right)n^2 + \tau\frac{\partial v_{k,i}}{\partial x}\left(\frac{\partial u^+}{\partial x} - \frac{\partial u^-}{\partial x}\right)n^2\,\mathrm{d}s. \tag{2.16}$$

2.4 KS Equation Solver Implementation

The present implementation needs to be able to support high-order elements for the generation of training data. To this end, the solution and test bases are Lagrange polynomials using Chebyshev node spacing. This will prevent spurious oscillations at element edges and resulting poor numerical conditioning.

Since the KS equation is fourth-order, it is expected to be extremely stiff. Explicit time stepping with such an equation quickly becomes computationally infeasible. Instead, we perform Newton's method at each time step to solve the unsteady system of equations. The linear solver uses the Generalized Minimal Residual Method (GMRES) [103]. This is an iterative solution technique that incrementally builds an orthonormalized Krylov subspace for the full system's Jacobian. At each iteration, the minimal residual solution in the partial Krylov subspace is found by QR decomposition. The QR decomposition is also performed incrementally via Givens rotations.

In GMRES, the full residual Jacobian matrix is not required, only the matrix-vector product. Our 1D DG method will form a large block tri-diagonal Jacobian that would be inefficient to form and store. For more practically sized problems, it is even infeasible to store the Jacobian in a sparse representation. These motivations lead to a class of finite-difference methods to compute the matrix-vector product [75, 20, 19]. Since the Jacobian matrix represents the

derivative of each residual entry with respect to each state, the matrix-vector product is a directional derivative. This leads naturally to a finite-difference technique, where the step is taken in the direction of the vector to approximate the matrix-vector product.

It has been observed that a first-order finite-difference is sufficient to accurately compute the matrix-vector product in most situations [75]. However, at least for the KS equation with our discretization, we observe this assertion is heavily dependent on the chosen step size. As opposed to the matrix-vector product with already orthonormalized Kyrlov vectors that form the Krylov subspace in GMRES, choosing a step size for full linear residual evaluation is more tricky. In this work we have used the step size presented by Yildirim *et al.* in [125]. This method is a variant of the one used by Brown and Saad in [20] which provides accurate results for differently scaled input vectors.

2.4.1 Verification

At this point the solver should be verified to ensure the output data are accurate. We will perform this verification in two ways. The first is to use the method of manufactured solutions to enforce an exact solution on the system using a source term. This method will test the convergence rate of the second- and fourth-order terms, since their implementation is relatively error prone. The second method will reproduce a series of bifurcations in the solution based on the parameterization of the equation. Specifically, the coefficients on all terms except the fourth-order diffusion term are held constant while the diffusion term is slowly weakened, leading to chaotic behavior.

In the method of manufactured solutions, the test equation is augmented with a source term s,

$$u_t + uu_x + u_{xx} + \nu u_{xxxx} + s = 0. \tag{2.17}$$

We can then assume a solution, which we take to be the Gaussian function

$$u = \exp\left(-2\left(x - \pi\right)^2\right) \tag{2.18}$$

on the domain $[0, 2\pi]$. Substituting Equation 2.18 into Equation 2.17 yields the necessary source term s. Plugging this source term back into Equation 2.17 gives an equation that has Equation 2.18 as its solution. We can then compare any numerical solution to the modified equation against the exact analytical solution to determine convergence.

In order to specify the solution, we require Dirichlet boundary conditions at either end of the domain. In the present implementation, these boundary conditions are weakly enforced by fixing the state and necessary derivatives outside the domain and calculating all fluxes normally. For the second-order term, only the state on either side is required to define the solution. For the fourth-order term the state and gradient on either side of the domain are required. Since we have the analytical solution to the modified equation, we can simply set these values to the exact solution at the boundaries. Higher derivatives are simply set to match the interior value.

Beginning with the second-order anti-diffusive term, for any polynomial order p we expect to converge at order $p + 1$ with mesh refinement [101]. For each test we measure the error with a continuous L^2-norm over each element and sum over all elements. Integration is performed using a 16-node Gauss-Legendre quadrature rule. Since convergence rates are only guaranteed asymptotically as mesh size approaches zero, we measure the convergence rate between the two finest meshes in each test. Applying the method of manufactured solutions we get the result in Figure 2.2. To two significant digits the expected convergence rate is achieved at each order. $p = 0$ is not tested because it is inconsistent under this discretization.

Moving on to the fourth-order diffusion term, we expect the convergence rates demonstrated by Georgoulis and Houston in [56]. The $p = 2$ discretization should converge at order 2, while

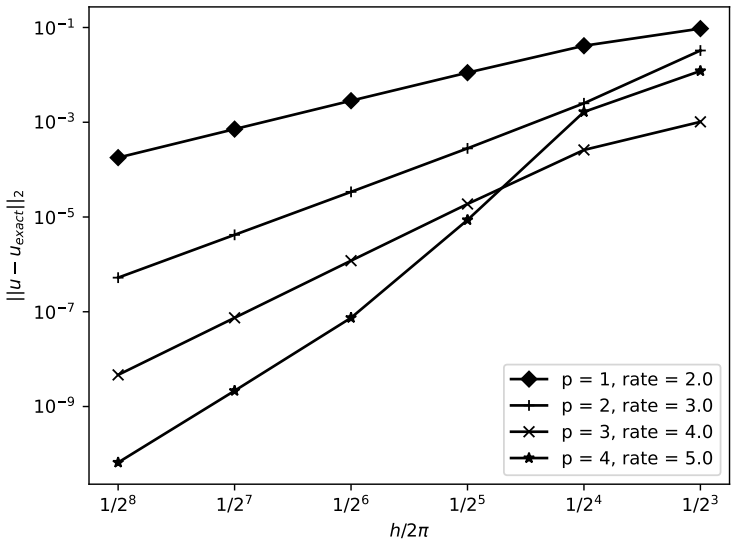


Figure 2.2: Second-order term achieves optimal $p+1$ convergence rate.

$p > 2$ solutions should converge at the optimal rate $p+1$. $p = 0$ and $p = 1$ are not considered because they are not consistent for this discretization. Testing proceeds identically to the second-order term, resulting in Figure 2.3. We can see the convergence rates are, once again, as expected to two significant digits.

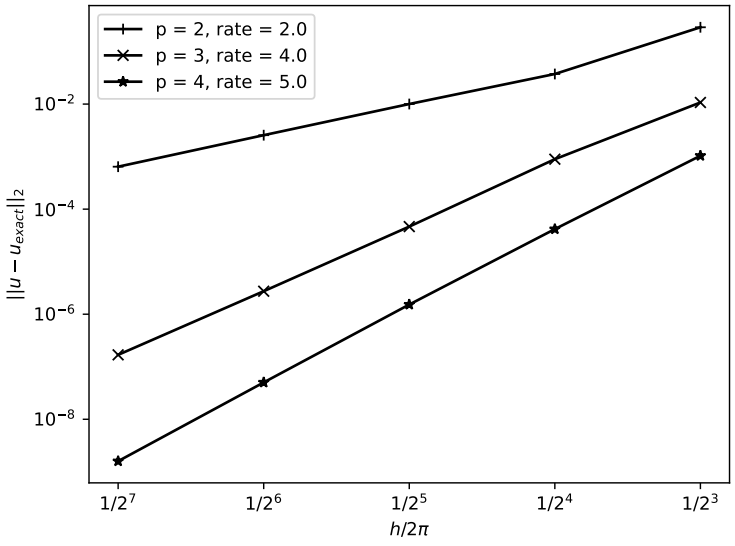


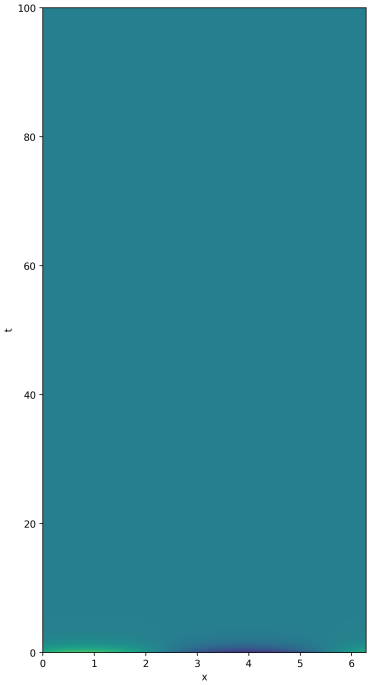
Figure 2.3: Fourth-order term achieves optimal $p+1$ convergence for orders except $p = 2$ where suboptimal convergence is expected.

Next, we perform a more stringent test on the behavior of the full KS equation implementa-

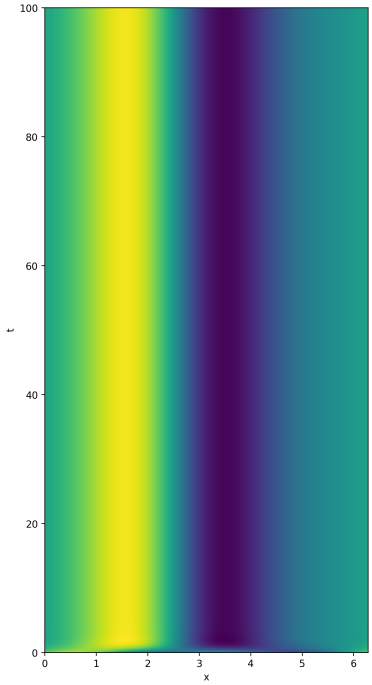
tion. We will attempt to reproduce the series of bifurcations demonstrated by Hyman and Nicolaenko in [67]. The character of KS equation solutions varies wildly with small changes in its parameterization. These solutions range from constant, to sinusoidal fixed points, to chaotic, based on the strength of the diffusion term.

While [67] simulates an equation with different time scaling, the character of the solutions should remain constant through the change. For this series of tests we only vary the strength of the fourth-order diffusion term. Since we want to be sure high-order solutions are correct for super-resolution model training, we perform this test at a relatively high order, $p = 7$, with only 16 elements. The domain is $[0, 2\pi]$, which makes the translation from Hyman and Nicolaenko's bifurcation parameter to the fourth-order viscosity coefficient convenient. The time stepping method is BDF2 with a time step of 10^{-2}. This level of time accuracy has proven able to capture and maintain the expected bifurcations, even at high spatial orders. Each test is run for 10,000 iterations for a simulation time of 100 units. (The three-stage, fourth-order diagonally implicit Runge-Kutta method of Crouzeix was attempted for these problems [27, 2], but little gain in computational time was observed relative to BDF2. Perhaps the implementation could be improved, or a better method chosen, for example the methods explored in [72].) The initial condition for all tests is $u_0(x) = \sin(x) + \cos(x) + \sin(2x) + \cos(2x)$.

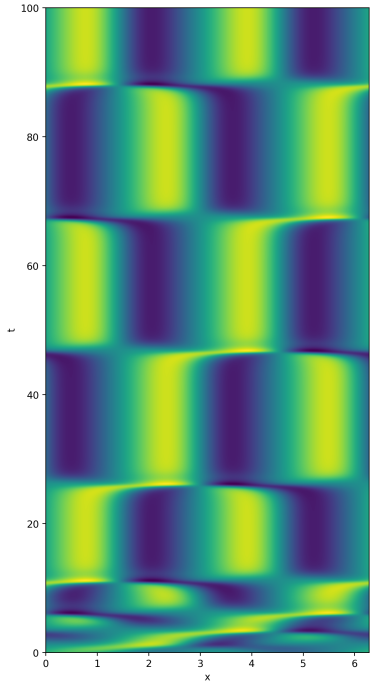
An example $x - t$ diagram for each bifurcation regime is shown in Figure 2.4. At the chosen spatial and temporal resolution, we are able to recover each bifurcation region at a viscosity setting within the expected range. It was observed that resolving the boundaries of each region is difficult at the chosen resolution. For example, the unstable bimodal fixed piont may decay to the stable fixed point too early. This behavior is expected; a quick exploration is not expected to resolve all bifurcation region boundaries exactly.



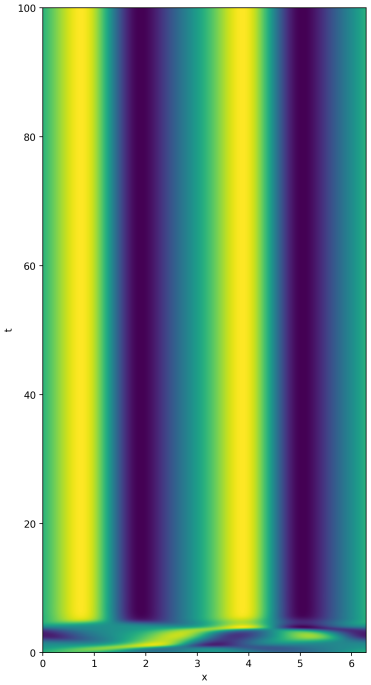
(a) $\nu = 2$:
global attractor at zero



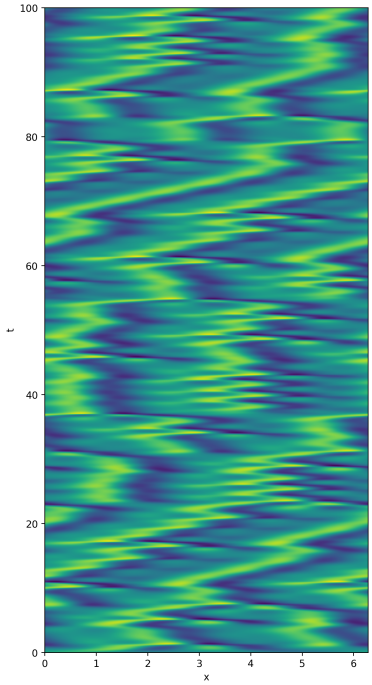
(b) $\nu = 0.4$:
unimodal fixed point



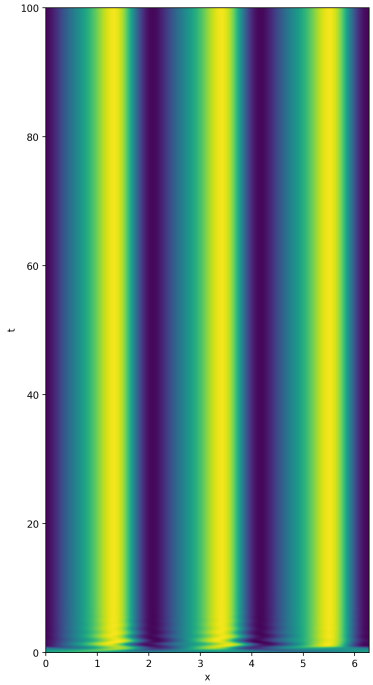
(c) $\nu = 0.2$:
unstable bimodal fixed point



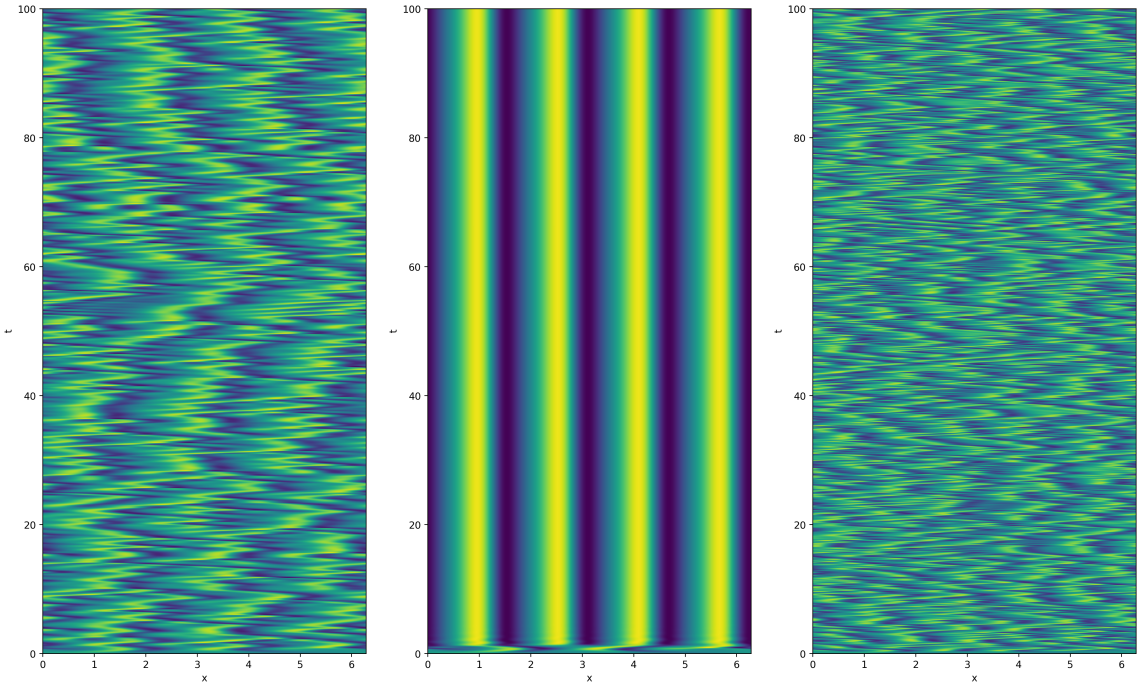
(d) $\nu = 0.16$:
stable bimodal fixed point



(e) $\nu = 0.08$:
oscillatory behavior



(f) $\nu = 0.0\bar{6}$:
trimodal fixed point



(g) $\nu = 0.0\bar{4}$: chaotic behavior

(h) $\nu = 0.04$: quadrimodal fixed point

(i) $\nu = 0.02$: chaotic behavior

Figure 2.4: Bifurcation regimes for DG solutions of the KS equation.

2.5 Summary

In this chapter we have presented governing equation discretizations. The Navier-Stokes equations will be used in Chapters 4 and 5 for training data generation and turbulent adaptation. A DG discretization for the KS equation was chosen. Extra time was spent to convert higher order terms from bilinear form into flux form for easy implementation. A DG KS equation implementation was developed and verified by checking the convergence of the implemented terms and reproducing the expected bifurcation regimes. The solver will be used for reconstruction testing in Chapter 3.

Chapter 3

Super-Resolution of the 1D Kuramoto-Sivashinsky Equation

3.1 Introduction to Super-Resolution Reconstruction

Super-resolution has been an active area of computer vision research for several decades, typically operating on digital image data. The goal is to reconstruct a high-resolution output from a low-resolution or distorted input. The down-scaling is typically posed as

$$\mathbf{u}_H = \mathcal{M}\left(\mathbf{u}_h\right), \tag{3.1}$$

where $\mathcal{M}$ is a, generally unknown, down-scaling operator that reduces the high-resolution input $\mathbf{u}_h$ to the low resolution output $\mathbf{u}_H$. Here, $\mathbf{u}_h$ and $\mathbf{u}_H$ represent field quantities over the spatial domain, *e.g.* a velocity in a CFD simulation. In super-resolution we seek an up-scaling model $\mathcal{M}^{-1}$ that approximates the original image as

$$\hat{\mathbf{u}}_h = \mathcal{M}^{-1}\left(\mathbf{u}_H\right), \tag{3.2}$$

where $\hat{\mathbf{u}}_h$ is the reconstructed image. Super-resolution is an ill-posed inverse mapping problem. Since exact solutions are generally impossible, we seek a solution that minimizes up-scaling error. Classical algorithms dominated in super-resolution for decades [39, 96]. With recent advances in compute capability and data availability, artificial neural networks have surpassed the performance of classical algorithms for super-resolution [122, 4]. Hence, they will also be the focus of this thesis.

Super-resolution is well-studied in the literature, with various network architectures tuned for different applications [122, 4]. A few of these architectures are most commonly used in super-resolution for fluids and are discussed here. Convolutional neural networks [80, 79] achieved state-of-the-art up-scaling performance when Dong *et al.* [35] formulated classical ideas in the form of a single convolutional neural network. Ledig *et al.* [81] increased performance by applying a generative adversarial framework to the problem [58]. Previous super-resolution work [38, 69] has also used residual networks [61], where a network stage does not learn the final output, but a correction to the input. These residual networks serve as the primary inspiration for the super-resolution reconstruction in this thesis.

Super-resolution in a fluid-dynamics context is typically applied not to a set of images, but to downsampled turbulent velocity fields. Fukami *et al.* [49] tested a hybrid convolutional neural network architecture on DNS data using max and average pooling for downsampling. Liu *et al.* [84] proposed an approach for spatio-temporal super-resolution using multiple convolutional paths, each handling different time ranges. Fukami *et al.* [50] proposed an alternative temporal super-resolution architecture built on their previous single-image work. Generative adversarial approaches have also been applied to super-resolution. Xie *et al.* [119] include an additional temporal discriminator during training to ensure the output's temporal coherence. Deng *et al.* [29] successfully resolve large wake fields with 4x and 8x upscaling factors using their generative adversarial approach.

Pradhan and Duraisamy [99] explored super-resolution for turbulent flows projected to dis-

continuous finite-elements. They showed remarkable reconstruction accuracy at relatively low polynomial orders. Xu, Pradhan, and Duraisamy [120] augmented the previous model by altering a hidden layer's weights with a trainable function of an input parameter. The excellent reconstruction of the baseline model was further improved.

Super-resolution, even in a fluids context, will typically use a down-sampling operator that convolves the input flow-field with a down-sampling kernel,

$$\mathbf{u}_H = \mathcal{M}\left(\mathbf{u}_h\right) = \mathbf{u}_h \otimes \mathcal{K}, \tag{3.3}$$

where $\otimes$ is a convolution operator and $\mathcal{K}$ is a blurring kernel. Our down-sampling model in DG is quite different. To down-sample a solution from a fine approximation space $\mathbf{u}_h$, we seek a solution in a coarse approximation space $\mathbf{u}_H$ that minimizes the difference between the two

$$\mathbf{u}_H = \arg\min_{\mathbf{u}_H} \int_{\Omega} \left|\mathbf{u}_h - \mathbf{u}_H\right|^2 \mathrm{d}\Omega. \tag{3.4}$$

This is a least-squares projection problem. After reducing this continuous form to a particular basis we find that our final down-sampling operation is

$$\mathbf{U}_H = \mathbf{M}_H^{-1}\mathbf{M}_h^H\mathbf{U}_h, \tag{3.5}$$

where $\mathbf{U}_h$ are the basis-function coefficients of state $\mathbf{u}_h$, and $\mathbf{U}_H$ are the basis function coefficients of state $\mathbf{u}_H$. The matrices $\mathbf{M}_H$ and $\mathbf{M}_h^H$ contain the integrals of each of the products of the $(p_H + 1)$ test functions multiplied by each of the $(p_H + 1)$ and $(p_h + 1)$ basis-functions, respectively. This down-sampling operator returns a least-squares optimal $\mathbf{u}_H$ in the form of its basis-function coefficients. The solution restriction is dependent on the input and target orders. In addition, there is little information about the original solution remaining. For example, a $p = 3$ solution with a tensor product basis has only four data points in each direction. Additional information from neighboring elements should help

alleviate this relative lack of data about the original projected state.

In the remainder of this chapter we will use the high-order DG implementation of the Kuramoto-Sivashinsky equation duscussed in Chapter 2 to generate data. High order data, at $p = 15$, is used to resolve the fine scales we are trying to reconstruct on relatively few elements. The coarse mesh resolution ensures projected solutions are poor, stressing the reconstruction technique. We will then move on to super-resolution reconstruction in Section 3.2.1. This section explores how super-resolution models behave when presented with various training and test datasets. We also show accurate reconstruction is possible with a relatively small model in 1D. This bodes well for the feasibility of the technique in higher dimensions.

3.2 Super-Resolution Reconstruction

Our ultimate goal is to perform mesh adaptation of statistically steady chaotic flows by varying element approximation order. We will use the KS equation solver to explore some properties of reconstruction and ensure the reconstruction technique is suitable for adaptation. KS equation solutions to chaotic parameterizations result in high-wavenumber variations in the solution. These variations can be accurately captured with a high-order discretization. When projected to lower orders, some variation will be lost and inter-element discontinuities will be introduced. This error in low-order states will provide ample opportunity to test reconstruction quality in one dimension before moving forward to more difficult reconstructions.

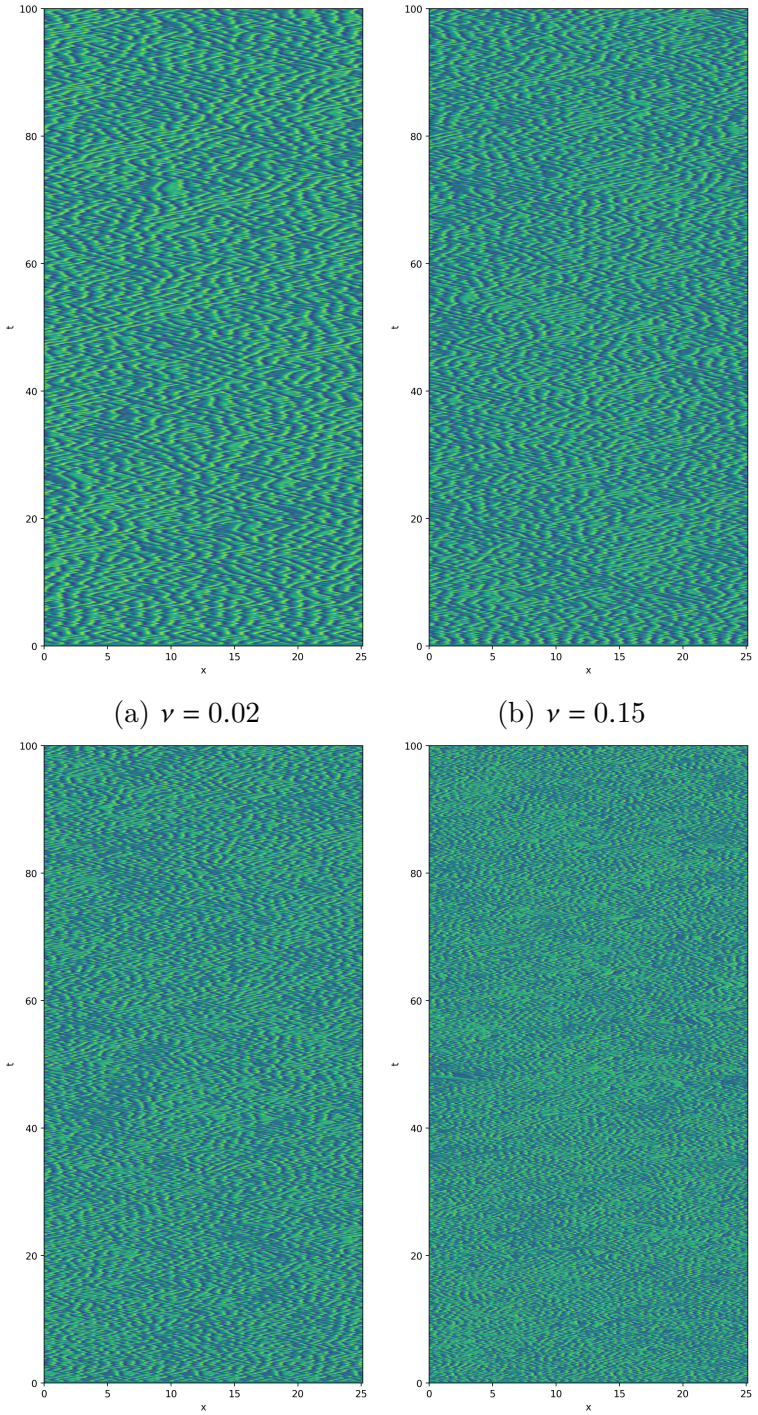
The data in this section will be used for both training and testing. The source data uses a high polynomial order, $p = 15$, to facilitate training and testing by projection to lower orders. The use of Chebyshev Lagrange nodes in the matrix-free nonlinear solver allows it to efficiently evolve the high-order discretization without spurious oscillations. Each dataset uses an identical spatial and temporal discretization; only the strength of the diffusion parameter is

varied. That spatial discretization is 32 evenly spaced elements at $p = 15$ over a domain of size 8π. This domain is four times larger than that of the bifurcation series in Figure 2.4 in order to improve low-wavenumber resolution in the spectral plots introduced later. Each dataset is evolved for 10,000 iterations with time step 10^{-2} using a BDF2 time discretization. The relatively low-order time-stepping scheme should lead to sub-optimal solver efficiency, however the temporal resolution is sufficient to generate and maintain solutions with the expected wavenumber variation. The $x-t$ diagrams of each dataset are shown in Figure 3.1. It is easy to see a significant wavenumber increase as the strength of the diffusion parameter is reduced. These wavenumber variations will be used to test the reconstruction model's ability to generalize across parameterizations, or lack thereof.

For the reconstruction model, we will train and test a fully connected artificial neural network. In the computer-vision field, neural networks have proven to be the tool of choice for super-resolution reconstruction in recent years [122, 4]. Here we will apply the technique directly to finite-element states by training a model to directly manipulate basis-function coefficients. The choice to act directly on basis-function coefficients means that the network is not basis independent. We will ensure this network architecture provides reasonable adaptations in Chapter 4. Alternative approaches, where the model effectively forms its own basis, are common in computer vision. Since data in computer vision are typically pixel values, finding the basis with the model makes sense. In our case we believe we have a reasonable basis to begin with and have not pushed the models further.

The model reconstructs one element at a time, taking neighboring element coefficients as input to ensure reasonable reconstruction continuity between elements. The input basis-function coefficients are normalized by subtracting the mean and dividing by the root-mean-square value of the element of interest. In our 1D case, the normalized input to the network F_{sr} is

$$F_{sr}\left(\frac{U_{H,l}-u_{m,c}}{u_{\mathrm{rms},c}}, \frac{U_{H,c}-u_{m,c}}{u_{\mathrm{rms},c}}, \frac{U_{H,r}-u_{m,c}}{u_{\mathrm{rms},c}}\right), \tag{3.6}$$



(a) $\nu = 0.02$ (b) $\nu = 0.15$

(c) $\nu = 0.01$ (d) $\nu = 0.005$

Figure 3.1: $x - t$ diagrams for reconstruction testing on KS equation data.

where each $U_{H,*}$ is the set of coarse (not reconstructed) basis-function coefficients for the left, l, center, c, and right, r, elements. The normalizing values are defined as:

$$u_{m,e} = \frac{1}{\|\Omega_e\|} \int_{\Omega_e} u_e \, \mathrm{d}x \qquad u_{\mathrm{rms},e} = \sqrt{\frac{1}{\|\Omega_e\|} \int_{\Omega_e} \left(u_e - u_{m,e}\right)^2 \, \mathrm{d}x}, \tag{3.7}$$

where Ω_e is the domain of element e, and u_e is its state. Instead of predicting output basis-function coefficients directly, the network predicts the required correction to the coefficients on the element of interest. Putting all this information together, the reconstructed state is found by

$$U_{h,c} = F_{sr}\left(\frac{U_{H,l} - u_{m,c}}{u_{\mathrm{rms},c}}, \frac{U_{H,c} - u_{m,c}}{u_{\mathrm{rms},c}}, \frac{U_{H,r} - u_{m,c}}{u_{\mathrm{rms},c}}\right) u_{\mathrm{rms},c} + U^H_{h,c}, \tag{3.8}$$

where $U_{h,c}$ are the high-order reconstructed basis-function coefficients, and $U^H_{h,c}$ are the fine-space basis-function coefficients representing the coarse state. This input normalization is based on the work of Pradhan and Duraisamy in [99].

The network is trained in a supervised setting. Both network inputs and target outputs are generated by least-squares projection of the original $p = 15$ data to lower orders. 1024 training samples are used from each dataset. This includes 32 solution snapshots with 32 elements each. Testing indicates more training samples than this have little influence on the final reconstruction quality. Since the samples need to cover a variety of conditions, it makes no sense to use states at adjacent time steps for training. Training snapshots are selected from every 25^{th} iteration starting at iteration 1000 in each dataset. Training proceeds using full dataset batches, the Adam optimizer [74], and a learning rate of 10^{-3}. Mini-batches were attempted to avoid local minima but showed little influence on final results. Additional validation samples are used to monitor the training process. Training proceeds until the validation loss stagnates or rises. This typically takes on the order of 1000 epochs.

Each test performs reconstruction on iterations 3000 to 10000 of the relevant dataset. This is to exclude the first 1000 iterations used to wash out the initial condition, the next 1000

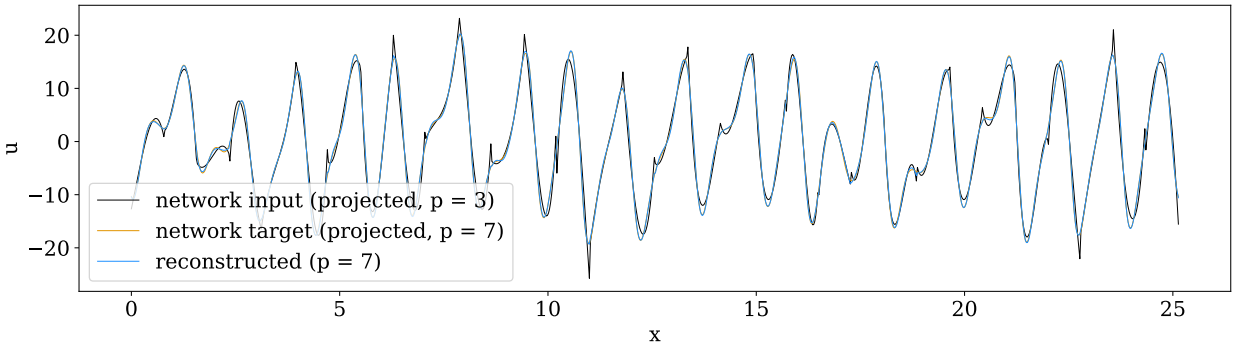
iterations used for training, and a final set of 1000 iterations used to separate training and test data. Performing inference on all 7000 states facilitates accurate averaged-energy spectra. The spectrum is computed using the Fourier transform of the two-point spatial autocorrelation function. Each spectrum is normalized by the value of the second mode to place to the approximately flat low-wavenumber region at approximately 1.

3.2.1 KS Reconstruction Across Equation Parameterizations

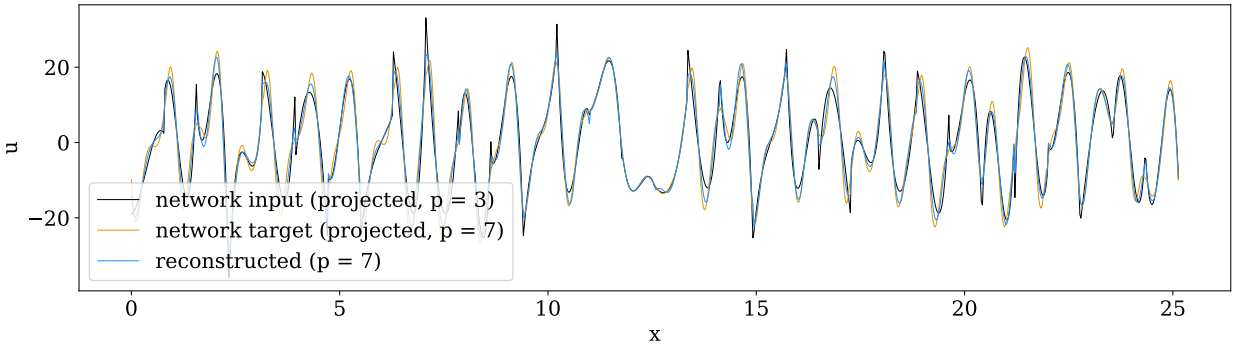
Our first reconstruction will be the simplest possible test of generalization. We train a network on the $\nu = 0.02$ dataset and test on unseen the $\nu = 0.02$ data from a much later time in the same simulation. The network is small, with only two hidden layers of 32 neurons each. Initial testing will be performed on this small network, additional testing with a larger network will follow. We will test reconstruction from $p = 3$ to $p = 7$. The results are shown in Figure 3.2. They consist of direct state comparisons between the network input, network target, and reconstruction. It also includes spectral comparisons between these three states.

When training and testing on the same equation parameterization, reconstruction is extremely accurate. The sample state comparison shows a nearly perfectly accurate reconstruction. The spectral result is also nearly perfect until it trails off in the high wavenumbers. The extra energy in the high wavenumbers comes from inter-element discontinuities. Performing a Fourier transform on these discontinuities is similar to the same operation on a square wave. Resolving corners requires the summation of ever higher wavenumbers. The results show those high wavenumbers have significantly decreased, approaching the correct, rapidly decaying, energy profile.

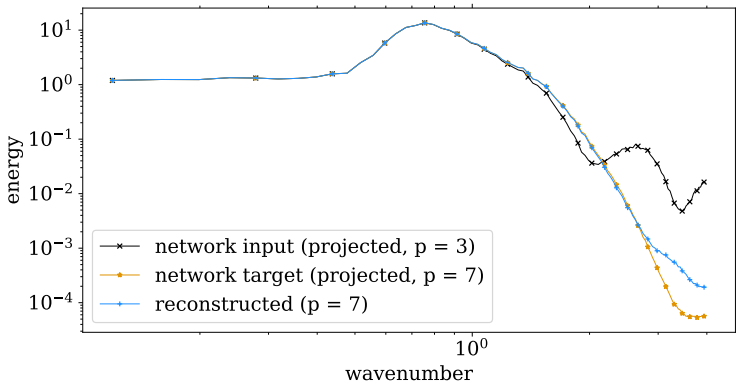
Figure 3.2 also tests the same network on higher wavenumber data, the $\nu = 0.01$ dataset. Visually comparing the reconstructed state with the target reveals more error than the previous case, when training and testing at the same viscosity. The reconstructed state clearly deviates from the network target throughout the snapshot. Including a wide range of



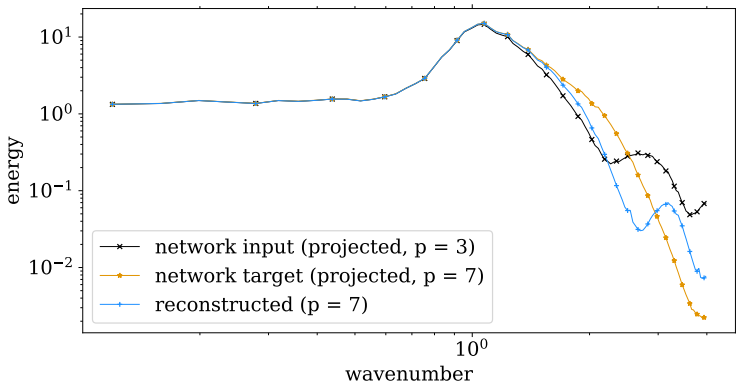
(a) $\nu = 0.02$, sample state reconstruction



(b) $\nu = 0.01$, sample state reconstruction



(c) $\nu = 0.02$, average spectrum



(d) $\nu = 0.01$, average spectrum

Figure 3.2: $p = 3$ to $p = 7$ reconstruction comparison for network trained on $\nu = 0.02$ data only.

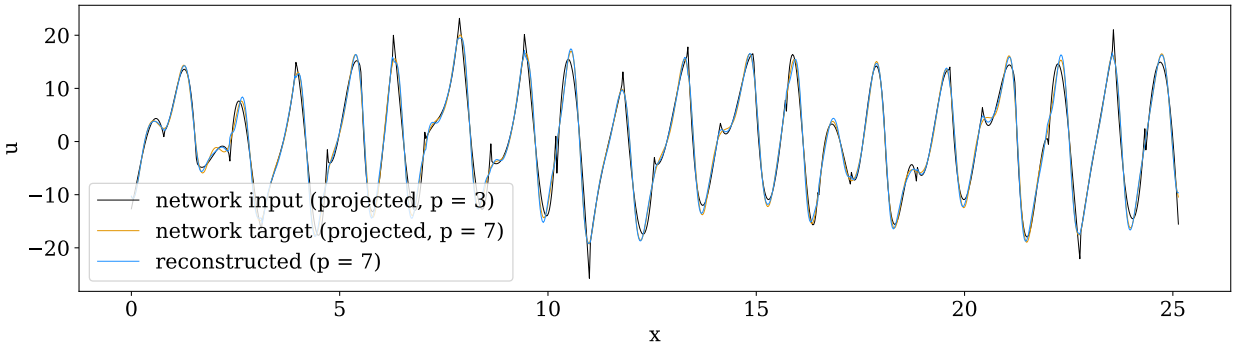
snapshots in the spectral comparison shows that the reconstruction attempt is approximately correct, but significantly off. Low wavenumbers move only slightly toward the target state. While high wavenumbers notably decrease, they do not follow the decay trend of the target state.

It is possible that the poor generalization of the network trained on low-wavenumber data to high-wavenumber data is directional. Meaning, a network trained on high-wavenumber data may generalize to low-wavenumber data, but not the other way around. We test this in Figure 3.3, where we train a network on $\nu = 0.01$ data and test on the same two datasets. Once again, the network is the same size, with two hidden layers of size 32.

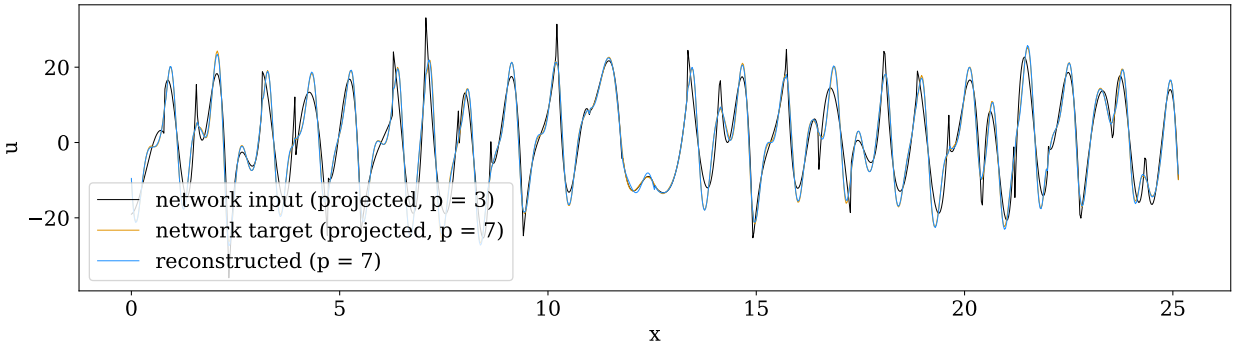
It appears that there is no generalization going in the other direction. The reconstructed $\nu = 0.01$ sample state is nearly exact, while the spectral reconstruction is also nearly exact. The exactness of the reconstruction at the same viscosity is even better than last time, except a spike in the very high-wavenumbers. This could simply be a coincidence due to random model weight initialization during training. Nonetheless, we get the expected result of accurate reconstruction when the parameterization of training and testing data match.

Moving back to the lower wavenumber $\nu = 0.02$ data is a completely different story. This time the reconstruction over-predicts nearly all wavenumbers. Note that the influence of the reconstruction only begins at the point where the network input and network target diverge. This is because the network is only designed to correct the input state, not reconstruct a new state from scratch. With this in mind, we can see in the $\nu = 0.02$ wavenumber plot that reconstruction deviation to the high side is immediate. Since the network was trained to expect higher wavenumbers, it appears to impose this expectation on the low-wavenumber data.

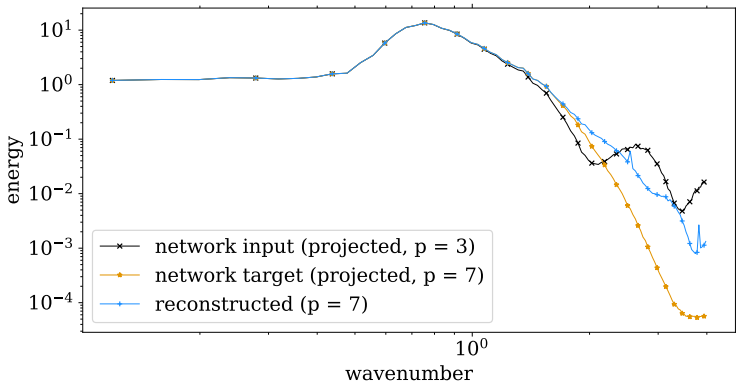
At this point we can be relatively sure that, at least under the current network input parameterization, the network will not generalize to unseen flow conditions. We should instead expect to train on a variety of equation parameterizations to expect accurate reconstruction



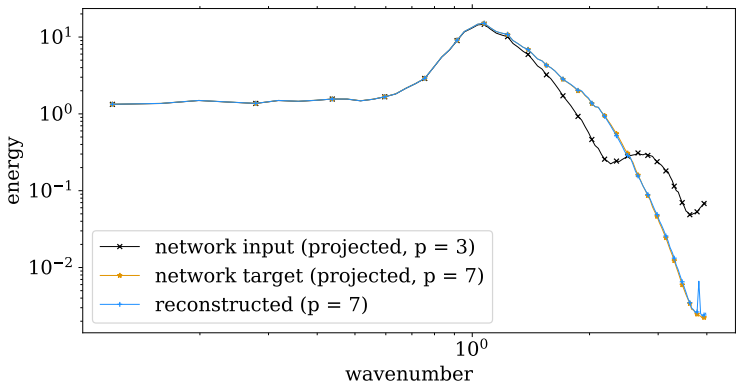
(a) $\nu = 0.02$, sample state reconstruction



(b) $\nu = 0.01$, sample state reconstruction



(c) $\nu = 0.02$, average spectrum



(d) $\nu = 0.01$, average spectrum

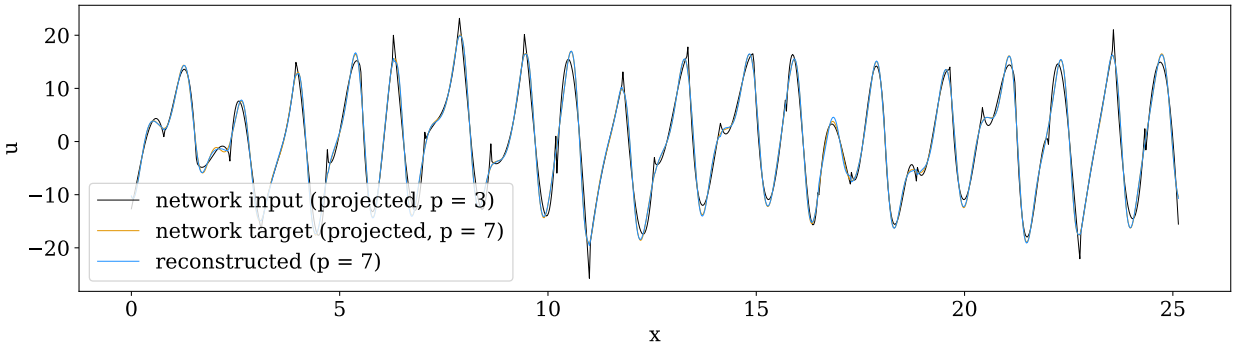
Figure 3.3: $p = 3$ to $p = 7$ reconstruction comparison for network trained on $\nu = 0.01$ data only.

on those parameterizations. Next, we will train on multiple parameterizations and check reconstruction performance on each.

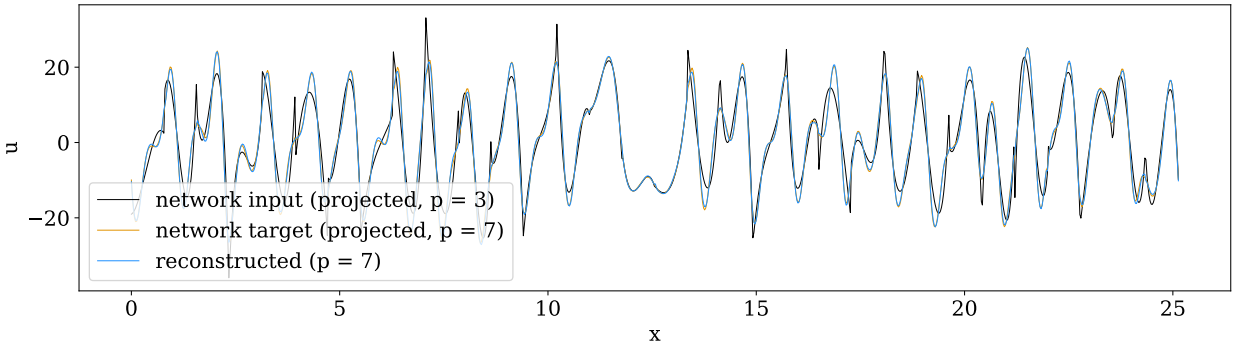
In Figure 3.4, the same small network is trained on $\nu = 0.02$ and $\nu = 0.01$ data. We are looking to see if the network is able to perform accurate reconstruction on unseen data of the same equation parameterization for which it was trained. The same, small, 32 neuron, two hidden layer network architecture is used. Note that the training datasets have simply been appended, so this network is trained on twice the data used in the previous examples. The number of training samples from each set remains the same.

Figure 3.4 shows accurate reconstruction on both datasets. The sample state reconstructions are nearly identical to the target state in both cases. The spectra are accurate as well, though slightly off in both cases. Notably, the low-wavenumber $\nu = 0.02$ case is off by more than the high-wavenumber $\nu = 0.01$ case. This could be due to the same phenomenon that lead to slightly more accurate reconstruction in Figure 3.3 than Figure 3.2. Also notable, once again, is the directional bias in reconstruction wavenumber. The low-wavenumber data reconstructs at a slightly too high-wavenumber. This may be expected due to the reduction of inter-element discontinuities as seen previously. But the high-wavenumber data reconstructs at a wavenumber that is consistently too low. This suggests that the reconstruction is somewhat biased toward a mix of the two wavenumbers on which it was trained. The two phenomena, reducing without eliminating spurious high wavenumbers and bias toward the average training data wavenumber, are consistent with the error magnitude between the two test sets. The low-wavenumber test set appears to exhibit higher error because the network is biased to reconstruct higher wavenumbers and inter-element discontinuities were not fully eliminated. The high-wavenumber test set appears to exhibit lower error because the tendency to produce lower wavenumber output cancels the tendency to not fully eliminate high-wavenumber errors.

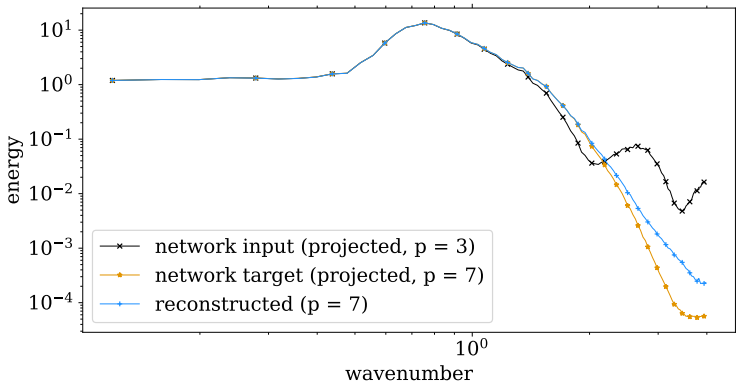
Next we will take the same network, trained on $\nu = 0.02$ and $\nu = 0.01$ data and move



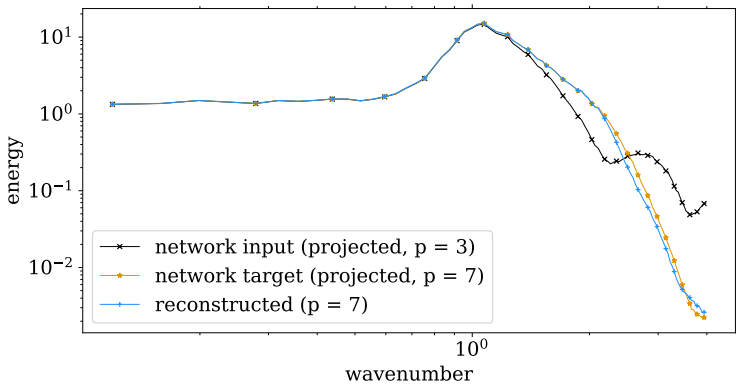
(a) $\nu = 0.02$, sample state reconstruction



(b) $\nu = 0.01$, sample state reconstruction



(c) $\nu = 0.02$, average spectrum



(d) $\nu = 0.01$, average spectrum

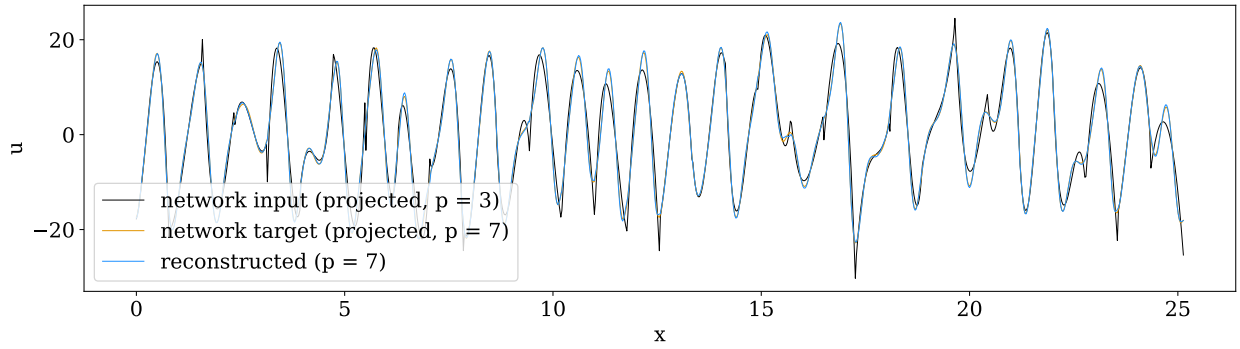
Figure 3.4: $p = 3$ to $p = 7$ reconstruction comparison for network trained on $\nu = 0.02$ and $\nu = 0.01$ data. Testing on unseen data at the training parameterizations.

on to unseen viscosities. The first test dataset contains $\nu = 0.015$ data, with intermediate wavenumvers between the training sets. The second test set contains $\nu = 0.005$ data, resulting in much higher wavenumbers than either of the training datasets. The first dataset is intended to test interpolation performance, and the second intended to test extrapolation performance.

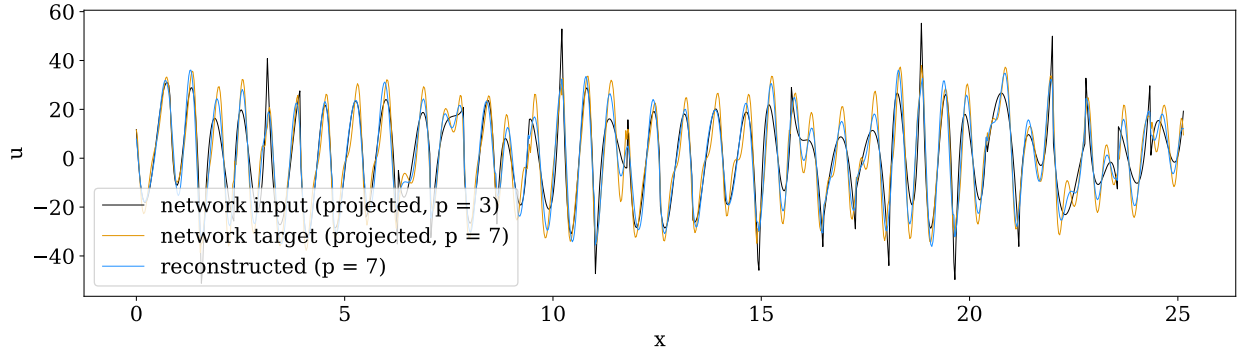
We should expect the interpolated parameterization to perform quite well. It has already been observed that the network appears to do some sort of interpolation between its training sets. This is because it reconstructs the low wavenumbers slightly too high, and the high wavenumbers slightly too low. Based on results we have already seen, we should expect the extrapolation to perform quite poorly, despite increased training data. Training on a single dataset and testing on another has performed poorly no matter which dataset was used for training.

The results of the parameterization interpolation and extrapolation test are shown in Figure 3.5. As expected, the $\nu = 0.015$ dataset reconstructs quite well. The spectral plot shows a slight high-wavenumber bias as seen in previous results. The qualitative comparison of reconstructed state appears nearly exact to the naked eye. On the other hand, the $\nu = 0.005$ dataset reconstructs rather poorly. Low-wavenumber reconstruction is relatively successful, matching the peak wavenumber of the target dataset. Beyond this point, the reconstructed wavenumber is too low relative to the target output. The sample state comparison also shows that the reconstruction is unable to capture the peaks and valleys of the target solution.

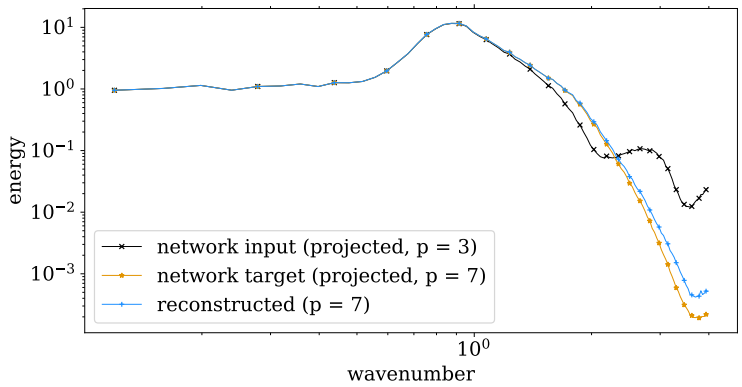
These results lay out that a super-resolution neural network, under the current input parameterization and architecture, should be able to accurately reconstruct multiple equation parameterizations. That is, as long as it has seen them in training, or the parameterization is interpolated by training data. This means that down the line, we should be able to use a single network across Reynolds numbers, and potentially across meshes. We will test that this observation holds in later chapters.



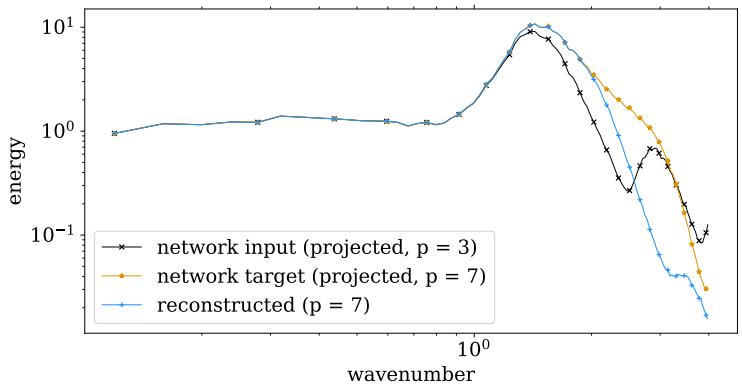
(a) $\nu = 0.015$, sample state reconstruction, interpolated parameterization



(b) $\nu = 0.005$, sample state reconstruction, extrapolated parameterization



(c) $\nu = 0.015$, average spectrum, interpolated parameterization



(d) $\nu = 0.005$, average spectrum, extrapolated parameterization

Figure 3.5: $p = 3$ to $p = 7$ reconstruction comparison for network trained on $\nu = 0.02$ and $\nu = 0.01$ data. Testing on unseen data at unseen parameterizations.

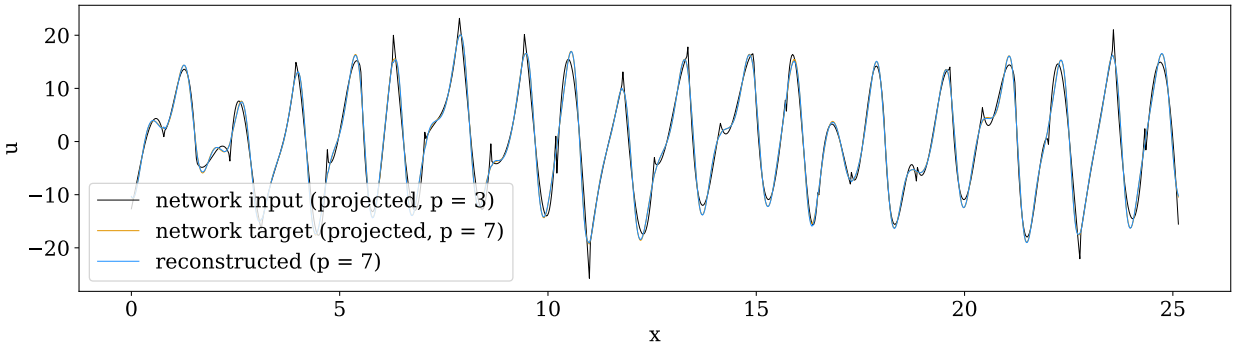
3.2.2 Influence of Network Size

A variable we have not changed so far is the size of the network. Thus far, we have stuck to a relatively small network of only 1,768 parameters. Even this small size has been able to produce accurate reconstructions on unseen data. This makes sense, given that the network input and output sizes for these $p = 3$ to $p = 7$ tests have been 13 and 8, respectively. 13 comes from four normalized basis-function coefficients on three elements, and a viscosity parameter. 8 comes from the 8 normalized coefficients of the $p = 7$ output. Regardless, we should test the performance of a larger network to see if we can improve the results. We will continue to enforce some regularization via early stopping to ensure the larger network does not over fit the training data. The test network will once again have two hidden layers, this time of 512 neurons each. This leads to a total of 273,928 parameters. The training set will be the mixed $\nu = 0.02$ and $\nu = 0.01$ datasets. The amount of training data is the same as for the smaller network. We repeat the mixed data tests with the small network.

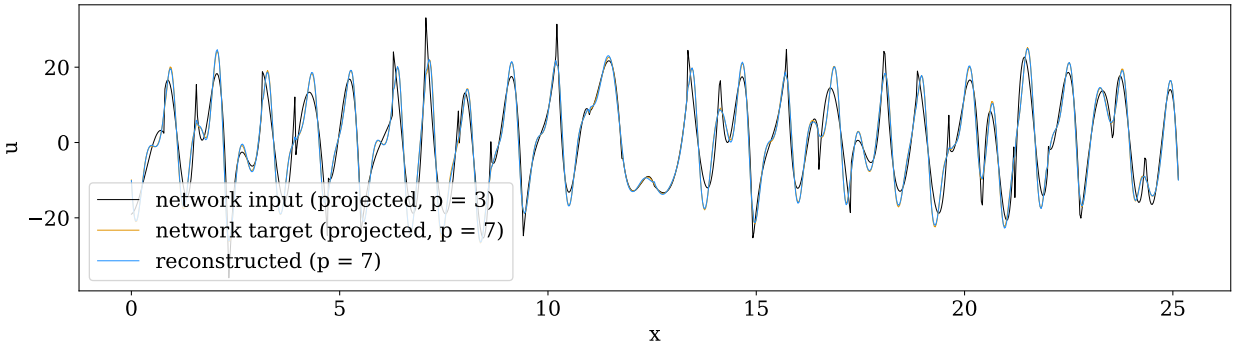
In Figure 3.6, we repeat the tests in Figure 3.4. The network is tested on unseen data with the same parameterization as its training set. We see that the character of the results remain the same, reconstruction is of high quality in both instances. That being said, the reconstruction quality for the larger network is clearly an improvement. The $\nu = 0.01$ data is nearly an exact match, while the $\nu = 0.02$ data is only slightly off. The error in the $\nu = 0.02$ data is still significantly larger than the $\nu = 0.01$ data, as it was with the smaller network. This could be due to the two phenomena mentioned previously, where the data averages its training set and high-wavenumber content from discontinuities is not completely removed.

Moving on to the extrapolation test in Figure 3.7, we see similar results to those in Figure 3.5. Reconstruction on the $\nu = 0.015$ dataset is perhaps only slightly improved. The $\nu = 0.005$ dataset shows almost no change at all with the larger network.

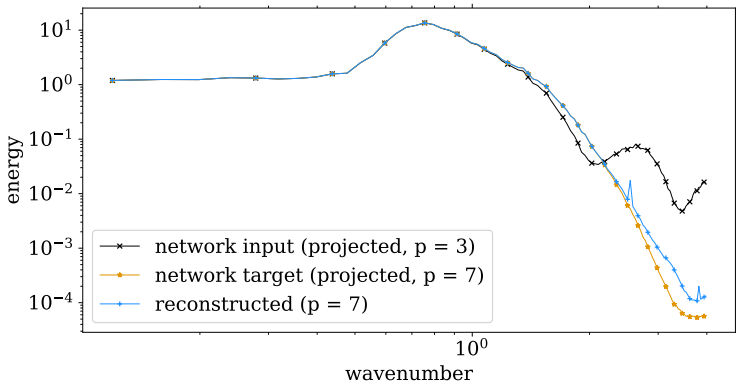
It appears the network size in the previous study was sufficient to capture phenomena of interest and set expectations for more strenuous reconstruction. This network architecture is



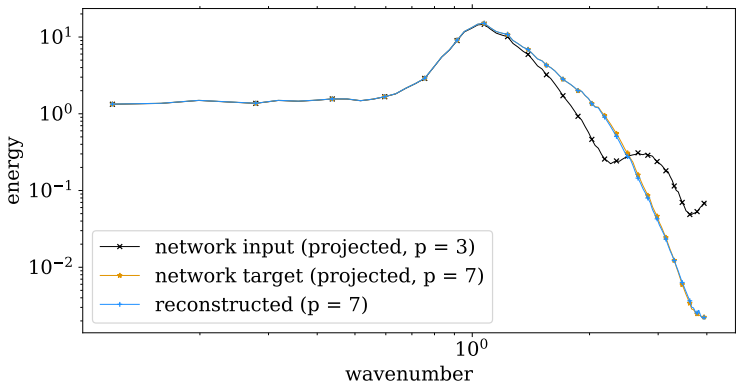
(a) $\nu = 0.02$, sample state reconstruction



(b) $\nu = 0.01$, sample state reconstruction

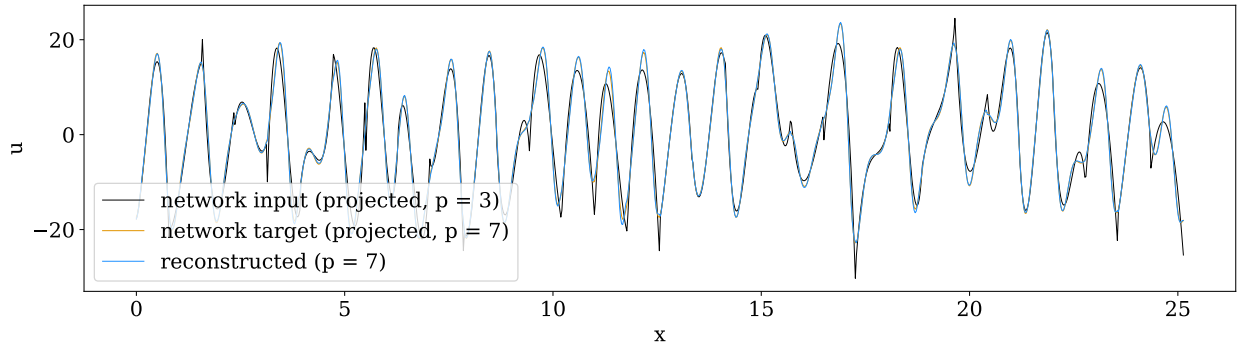


(c) $\nu = 0.02$, average spectrum

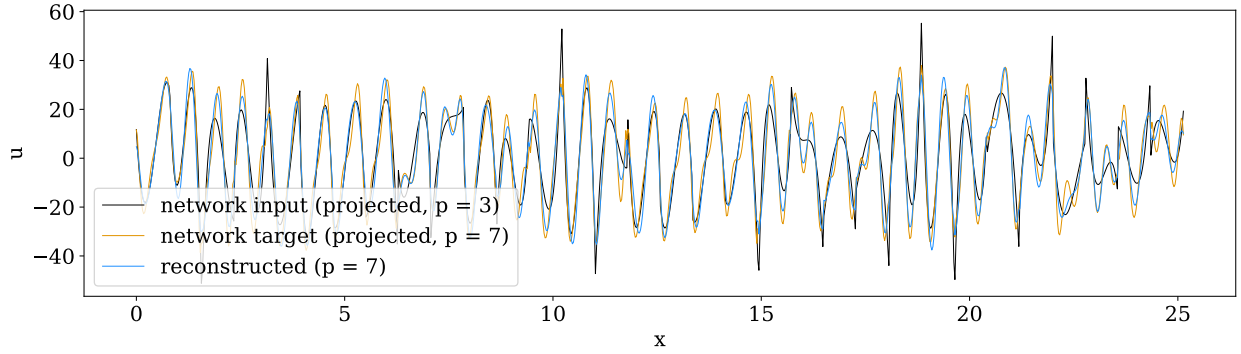


(d) $\nu = 0.01$, average spectrum

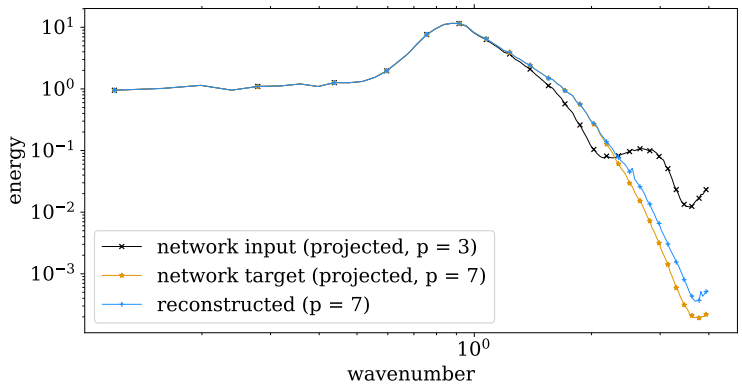
Figure 3.6: $p = 3$ to $p = 7$ reconstruction comparison for a large network trained on $\nu = 0.02$ and $\nu = 0.01$ data. Testing on unseen data at the training parameterizations.



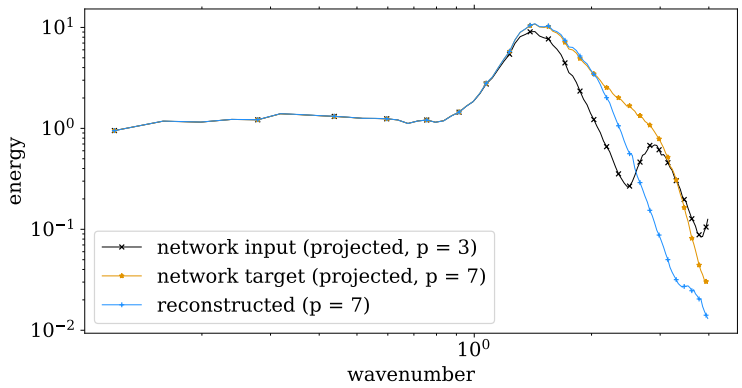
(a) $\nu = 0.015$, sample state reconstruction, interpolated parameterization



(b) $\nu = 0.005$, sample state reconstruction, extrapolated parameterization



(c) $\nu = 0.015$, average spectrum, interpolated parameterization



(d) $\nu = 0.005$, average spectrum, extrapolated parameterization

Figure 3.7: $p = 3$ to $p = 7$ reconstruction comparison for a large network trained on $\nu = 0.02$ and $\nu = 0.01$ data. Testing on unseen data at unseen parameterizations.

sufficient to reconstruct unseen data at the same equation parameterizaiton. We can expect reconstructing data at unseen parameterizations to have poor performance. As long as the test data interpolates the training set, we should expect a single network to perform accurate reconstructions on unseen data on parameterizations across its training set.

3.3 Summary

In this chapter we have implemented and verified a solver for the KS equation. This solver is designed to support high-order elements. Training data at $p = 15$ was generated to perform various tests of super-resolution reconstruction. It was shown that the reconstruction models perform well on data outside of their training set. This only applies when the underlying data was generated at a similar equation parameterization. Changing the equation parameterization significantly changes the character of the solution causing reconstruction to fail. Adding training data at a variety of viscosities fixed this problem for the KS equation. Testing on data between the training set viscosities showed promise. Extrapolating beyond the training parameterizations proved futile.

Chapter 4

Super-Resolution Reconstruction in Higher Dimensions

4.1 Super-Resolution in Two Dimensions

4.1.1 Methodology

Data for training and testing is projected from a high-resolution turbulent channel flow large-eddy simulation (LES). We simulate a $Re_\tau \approx 395$ plane turbulent channel flow using *eddy*. An example snapshot is shown in Figure 4.1. The channel dimensions are 2π x 2 x π in the streamwise, wall-normal, and spanwise directions, respectively. The setup follows that of Moser, Kim, and Mansour [89]. A constant streamwise body force maintains turbulent flow at the specified Reynolds number. *eddy* is efficient at high-orders, and only 8 x 12 x 8 elements at $p = 15$ are required for a well-resolved LES. This simulation data could be projected directly to lower orders to form the training and testing data. Such a decision would, however, limit the size and flexibility of our tests to a single element distribution and size. For this paper we also restrict our focus to 2D elements, so some sort of projection will

Parts of this chapter appear in or are adapted from our previously published paper [85].

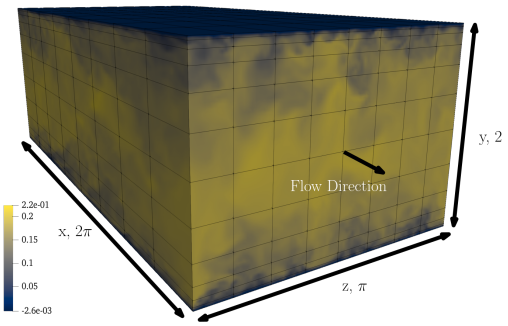


Figure 4.1: x (streamwise direction) momentum contours of an original channel flow snapshot used for training and testing. High-order $p = 15$ element boundaries are shown.

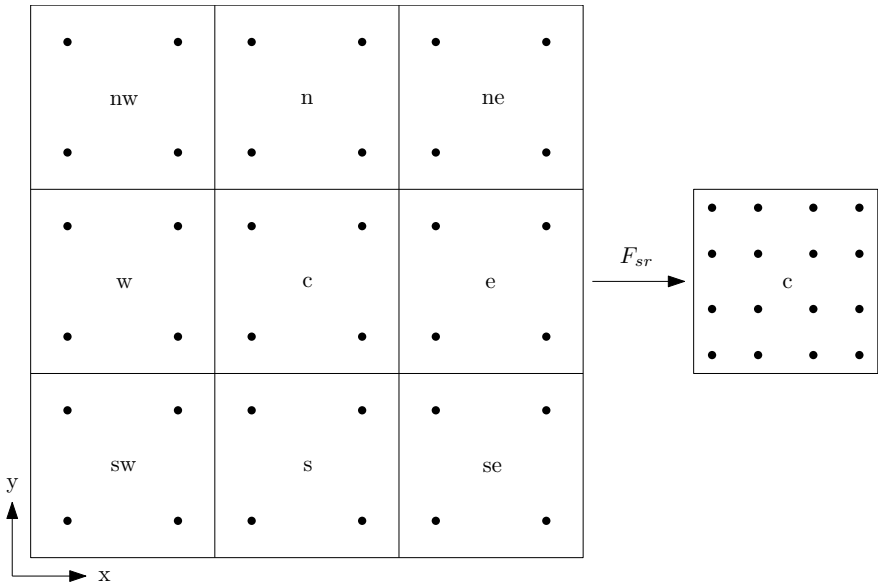


Figure 4.2: $p = 1$ to $p = 3$ super-resolution of element "c" using neighboring element data. Dots indicate Lagrange node positions for each degree of freedom.

be required from the originally 3D elements. We have chosen to sample the channel flow at a regular grid of 512 x 256 x 512 points in the streamwise, wall-normal, and spanwise directions, respectively. We use these data to perform least-squares projection to 2D DG data at any chosen slice and mesh resolution. Our elements use tensor-product Lagrange polynomials with Chebyshev node spacing.

Our networks directly input and output DG basis function coefficients. In general, the network could use the entire flow domain to predict state on a single element, but this is not computationally feasible. Instead, we focus on a neighborhood of elements around the

super-resolution target element as shown in Figure 4.2. We restrict our attention to only momentum coefficients in the streamwise and spanwise directions. Since our flow is nearly incompressible, the density is nearly constant at one, and the momenta are nearly identical to velocities. Testing proves that the basis function coefficients must be normalized for performance. We follow Pradhan and Duraisamy [99] for our normalization by defining the mean and root mean square (r.m.s.) values on an element for each rank as

$$\mathbf{u}_{m,e} = \frac{\int_{\Omega_e} \mathbf{u}_e \, \mathrm{d}\Omega}{|\Omega_e|} \tag{4.1}$$

and

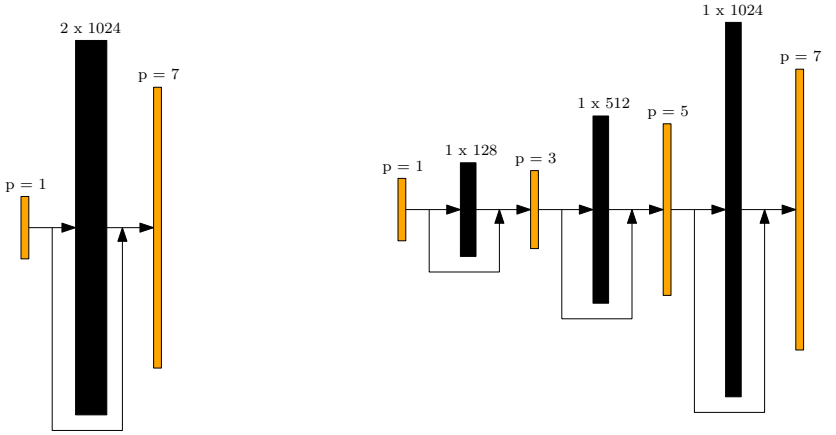
$$\mathbf{u}_{\mathrm{rms},e} = \sqrt{\frac{\int_{\Omega_e} \left(\mathbf{u}_e - \mathbf{u}_{m,e}\right)^2 \, \mathrm{d}\Omega}{|\Omega_e|}} \tag{4.2}$$

where $\mathbf{u}_e$ is the state of the streamwise and spanwise momentum components on element e and $|\Omega_e|$ is the volume of element e. The mean and r.m.s. quantities are used to normalize each rank of the input basis function coefficients. The final network input is formed by concatenating the normalized coefficients of the central element with its neighbors

$$F_{sr}\left(\frac{\mathbf{U}_{H,c} - \mathbf{u}_{m,c}}{\mathbf{u}_{\mathrm{rms},c}}, \frac{\mathbf{U}_{H,n} - \mathbf{u}_{m,c}}{\mathbf{u}_{\mathrm{rms},c}}\right) \tag{4.3}$$

where $(\cdot)_c$ denotes central element quantities and $(\cdot)_n$ denotes quantities on all neighboring elements. Note that neighboring elements are still normalized by the central element's mean and r.m.s. values. This is because the mean and r.m.s. can vary significantly from one element to the next, so that normalizing by different values causes unnecessary discontinuities in the input data.

Our networks do not predict a final output state. Instead they predict the correction to the input central state normalized by the input r.m.s. value. This setup ensures that the network predicts only velocity differences, which has proved an effective technique for models in domains from semantic segmentation to super-resolution. The final network input and



(a) Example fully connected single-shot network architecture to super-resolve $p = 1$ to $p = 7$.

(b) Example incremental network to super-resolve $p = 1$ to $p = 7$ with up-scaling to intermediate resolutions $p = 3$ and $p = 5$.

Figure 4.3: Examples of fully connected and incremental super-resolution architectures. Black boxes represent fully connected networks labelled [hidden layers x neuron count]. Orange boxes represent input and output state at the indicated order. Direct connections from input to output represent the addition of the original state to the network output.

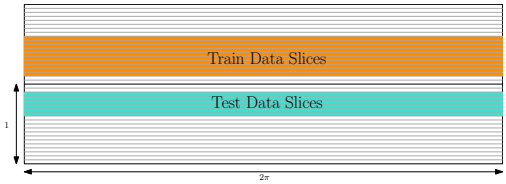


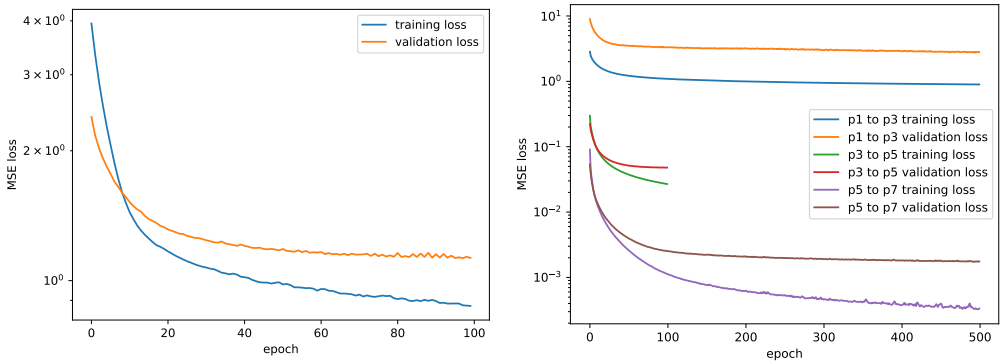
Figure 4.4: Training and testing sample slice locations in turbulent channel data.

output are then

$$\mathbf{U}_{h,c} = F_{sr}\left(\frac{\mathbf{U}_{H,c} - \mathbf{u}_{m,c}}{\mathbf{u}_{\text{rms},c}}, \frac{\mathbf{U}_{H,n} - \mathbf{u}_{m,c}}{\mathbf{u}_{\text{rms},c}}\right)\mathbf{u}_{\text{rms}} + \mathbf{U}^{H}_{h,c} \tag{4.4}$$

where $\mathbf{U}^{H}_{h,c}$ denotes the coarse state on the central element projected into the fine space. We assume the fine space contains the coarse space making the projection operation lossless. The prolongation operation is simply Equation 3.5 with the fine and coarse spaces reversed with the appropriate modification to the number of test functions.

To form the F_{sr} network, the most obvious option is to construct a fully connected artificial neural network as shown in Figure 4.3a. We construct this network with ReLU activation



(a) History for single shot $p = 1$ to $p = 7$ network.

(b) History for each network used in $p = 1$ to $p = 7$ super-resolution.

Figure 4.5: Example training histories.

functions on all layers except the last [93]. This option performs well for small jumps in polynomial reconstruction order. However, we find that for large order jumps, performance is poor. To fix this issue we propose a second network architecture where the reconstruction over large order jumps is incremental as shown in in Figure 4.3b. Effectively, we use several independently trained fully connected networks, each network trained for a particular order jump. Each network computes a normalized correction to the input state as described above, so each incremental network will predict increasingly finer flow features.

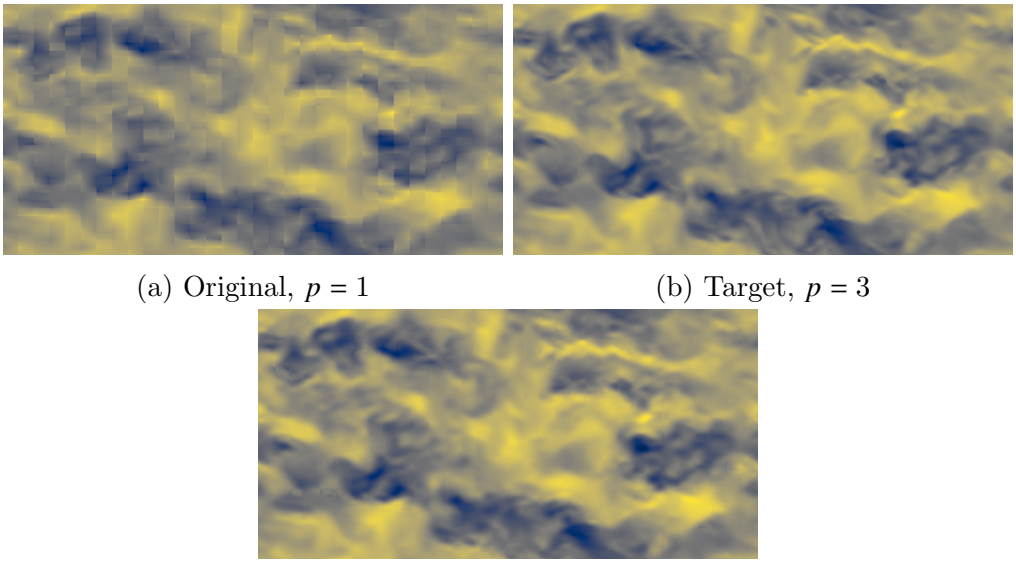
Each network, for both single-shot and incremental designs, is trained in a supervised framework with input / output pairs each projected from the original LES data at different orders. Training and testing data are taken from a region relatively close to the center of the channel. This ensures the character of the flow is nearly homogeneous isotropic with few wall effects. We will treat the generalization to near-wall flows in future work. The regions covered by training and testing are shown in Figure 4.4. While we have 256 sampled planes in the wall-normal direction, we do not use every subsequent plane for training and testing. There is an eight plane gap between each training and testing plane to ensure the data are not too similar between planes. We use the top side of the channel to gather training data and the bottom side for testing. We take care to ensure data near the center are not used for training or testing because the input data would be similar between those parts of the training and

testing sets. Training data are split 80%, 20% into training and validation sets, respectively. Training uses the Adam optimizer [74], a mean squared error loss, and a learning rate of 1×10^{-4} until the validation loss stops decreasing. This usually takes on the order of 100 epochs. Example training and validation loss histories are shown in in Figure 4.5.

4.1.2 Results

Our testing framework can choose any resolution less than or equal to the sampled LES resolution for testing. For all tests that follow, the mesh resolution, in terms of elements per direction, is chosen such that the number of degrees of freedom per direction is the same as the original simulation. This keeps the tests roughly in line with the classical notion of super-resolution, where one attempts to resolve the "true" image from a distorted or down-scaled image. Note that because we are working with DG, holding the number of degrees of freedom per direction constant is not quite sufficient to recover the original flow-field. This is because high-order elements are able to represent a solution more accurately with fewer degrees of freedom and the original simulation was run at a relatively high-order $p = 15$. As a result, the target values in the following tests will deviate from the original spectral content at high-frequencies.

Our primary target in the following tests will be to recover the spectral content of the target flow-field. When looking at a solution's spectral content, one will typically neglect the relatively low-energy high-frequency modes. In our situation, those high-frequency modes are beyond the frequencies representable by the polynomial solution on the elements. Instead, the high-frequency content is a result of discontinuities between elements. An accurate per-element up-scaling of a complete DG flow-field should take into account each element's neighbors and not introduce spurious discontinuities between elements. To this end we seek to reduce low and high-frequency spectral errors in the following tests. To approximately distinguish between the two frequency regimes, we define a Nyquist spatial frequency $f_{nq,d}$



(a) Original, $p = 1$ (b) Target, $p = 3$

(c) Super-resolved, $p = 3$

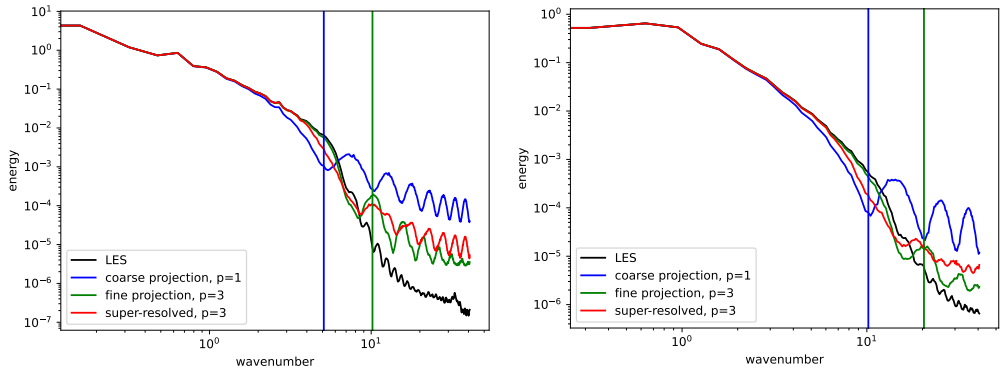
Figure 4.6: $p = 1$ to $p = 3$ super-resolution test on 32 x 32 elements with a single hidden layer fully connected network of 128 neurons. Streamwise velocity contours at $y^+ \approx 247$.

for each direction d as

$$f_{nq,d} = \frac{1}{2} \frac{N_{e,d} \, (p+1)}{L_d} \tag{4.5}$$

where $N_{e,d}$ is the dumber of elements in direction d, all elements are of polynomial order p, and L_d is the domain length in the d direction. This cutoff will be marked by a vertical bar in the spectral plots.

We begin by super-resolving a projected $p = 1$ field to $p = 3$. The network is a single hidden layer fully connected network with hidden layer size 128, similar to Figure 4.3a with different size parameters. We use 10 LES snapshots and 8 planes from each snapshot for training. Discretizing each plane into 32 x 32 elements and reserving 20% of the sample for validation leaves 65,536 training samples. Testing uses three planes from the opposite side of the channel. A spectrum is taken for each test plane on each snapshot, the spectra are then averaged over the 10 snapshots to reduce noise. The results are shown in Figures 4.6 and 4.7. The super-resolved flow-field is qualitatively almost identical to the target field. The spectral content shows identical low frequencies, and nearly matching middle and high-



(a) Streamwise spectrum

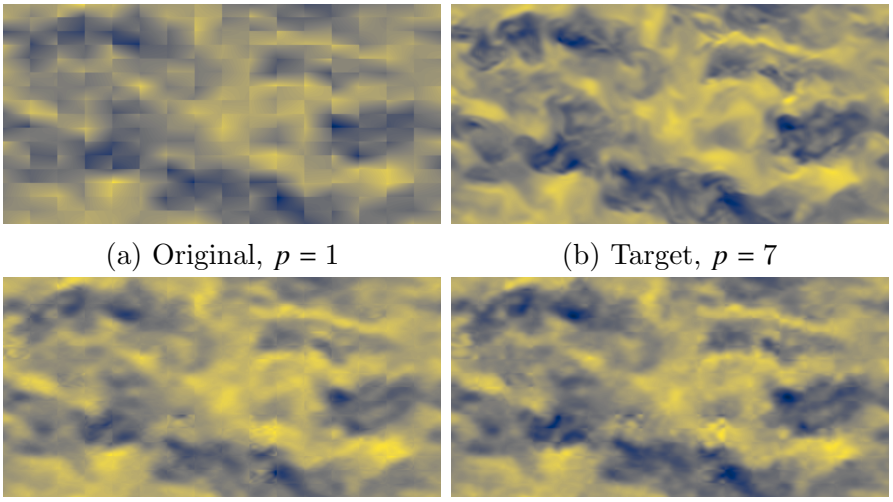
(b) Spanwise spectrum

Figure 4.7: Streamwise and spanwise energy spectra for $p = 1$ to $p = 3$ super-resolution on 32 x 32 elements.

frequency content for the streamwise spectrum. The spanwise spectrum is somewhat less accurate but still a significant improvement over the input state.

With the apparent success of $p = 1$ to $p = 3$ super-resolution with a relatively small network, we are able to increase the difficulty. We repeat the experiment for $p = 1$ to $p = 7$ super-resolution, once again with a fully connected neural network, this time with hidden layer size 512. Once again we use the same 10 snapshots, 8 planes each for training, 3 planes for testing. Since we are using the same amount of input data with a higher target order, we have more information per sample and the number of samples decreases. Instead of projecting to 32x32 elements, we project each slice to 16x16 elements. This results in a total of 16,384 training samples, once again with 20% having been reserved for validation. With this harder problem the results have degraded significantly. Qualitatively, the resulting field lacks high-frequency content and maintains some inter-element discontinuities as shown in Figure 4.8c. These observations are backed up by the spectral content shown in Figure 4.9. Increasing the width and depth of the single-shot network fails to improve the solution in testing, indicating that the original network is sufficiently sized.

We repeat the $p = 1$ to $p = 7$ test once again, this time with the incremental super-resolution network. The training and test samples are exactly the same as in the single-shot case. The



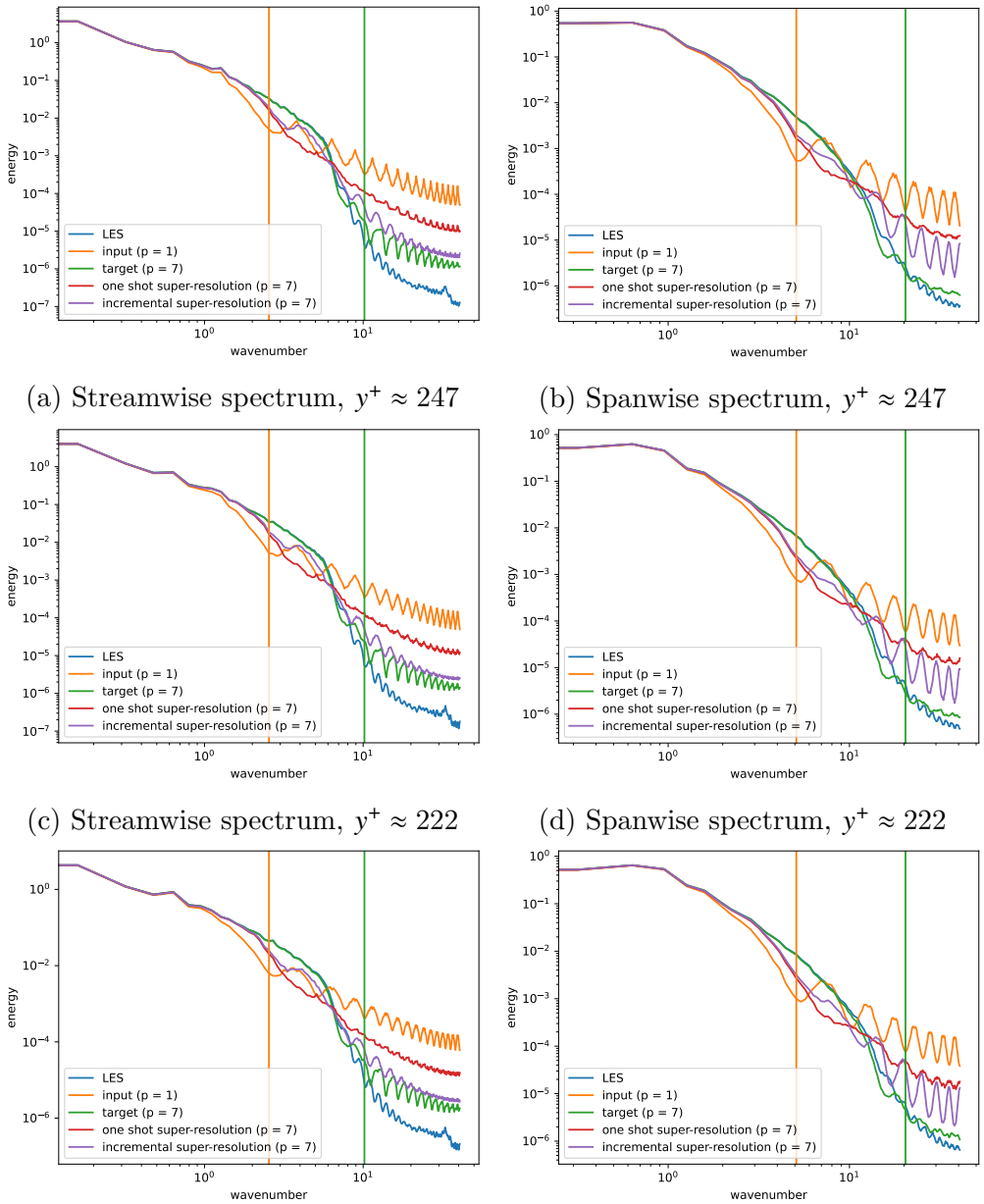
(a) Original, $p = 1$ (b) Target, $p = 7$

(c) One shot, $p = 7$ (d) Incremental, $p = 7$

Figure 4.8: Comparison of single-shot and incremental super-resolved fields with the input and target fields. Streamwise velocity contours at $y^+ \approx 247$.

successive networks each contain a single hidden layer with sizes 128, 512, and 1024. Just as in the single-shot network case, each network is trained until its validation loss stagnates. Immediately, we see a qualitative improvement in the results of Figure 4.8: the flow has more high-frequency content while also reducing inter-element discontinuities. The spectral content in Figure 4.9 confirms the qualitative observations. The spectral improvement also holds across testing planes, this should be expected, since we are far from the walls of the channel, the flow should be of similar character at all test planes. The spanwise spectral content does not appear to be as improved as the streamwise spectral content. This may be due to the fact that the input data has the same number of degrees of freedom in the streamwise and spanwise directions despite the streamwise direction being twice as long. The spanwise direction is therefore able to represent higher frequencies which may be more difficult to capture.

Since the incremental super-resolution method computes increasingly high-order corrections as a state makes its way through the network, it may be of interest to see which spectral frequencies are affected by each step. To this end, in the case of $p = 1$ to $p = 7$ super-resolution with order increments of two, one would expect the lowest frequencies to be



(a) Streamwise spectrum, $y^+ \approx 247$

(b) Spanwise spectrum, $y^+ \approx 247$

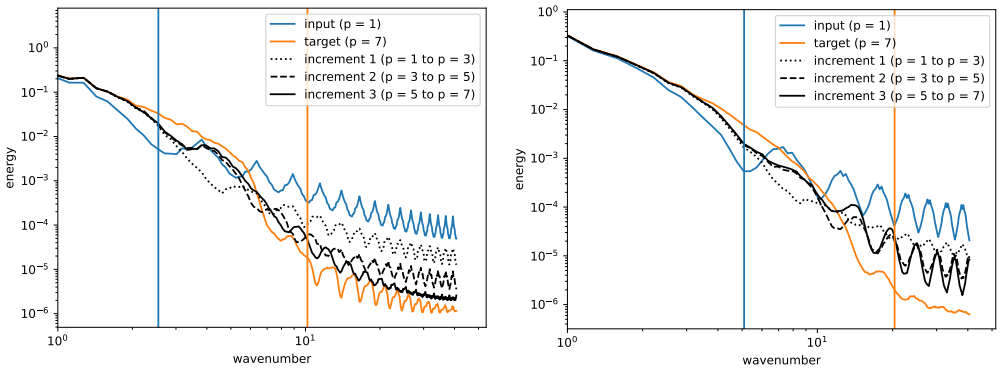
(c) Streamwise spectrum, $y^+ \approx 222$

(d) Spanwise spectrum, $y^+ \approx 222$

(e) Streamwise spectrum, $y^+ \approx 198$

(f) Spanwise spectrum, $y^+ \approx 198$

Figure 4.9: Streamwise and spanwise energy spectra comparison for $p = 1$ to $p = 7$ super-resolution.



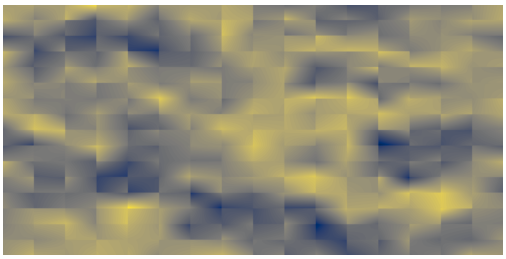
(a) Streamwise spectrum

(b) Spanwise spectrum

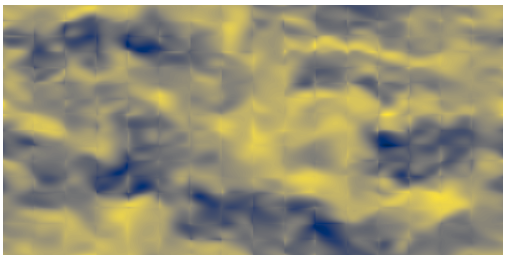
Figure 4.10: Spectral progression of $p = 1$ to $p = 7$ incremental super-resolution.

handled first, then increasingly high frequencies as the network progresses. Figure 4.10 shows the spectral progression through the iterations of super-resolution for the streamwise and spanwise spectra. In the streamwise case it appears that the initial hypothesis is roughly true below f_{nq} of the $p = 7$ state. It also appears that each increment reduces the level of inter-element discontinuity as shown by the decreasing energy content above f_{nq}. The spanwise spectrum is once again less impressive than its streamwise counterpart but the general character of our observations holds.

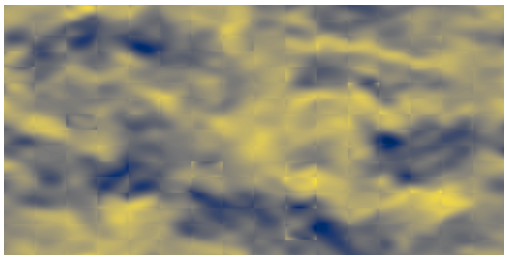
Figure 4.11 shows the progression of fields predicted by incremental super-resolution corresponding to the spectra in Figure 4.10. Each row compares the least-squares projected target state with the reconstruction at the corresponding stage. The overall spectral features shown in Figure 4.10 are apparent. While inter-element discontinuities are clearly noticeable in the initial condition, they are mostly removed by the second incremental reconstruction. Lower frequency content is also clearly added at each step, but some details are revealed that are masked by the averaging of the spectral plots. In the first reconstruction step, the noise added appears very similar to the truth case, though the result is clearly lacking some higher-frequency content. By the second reconstruction step, while the reconstruction remains mostly coherent, oscillations are apparent in some elements. This oscillatory behavior appears to carry through and increase in the third iteration. These oscillations add middling



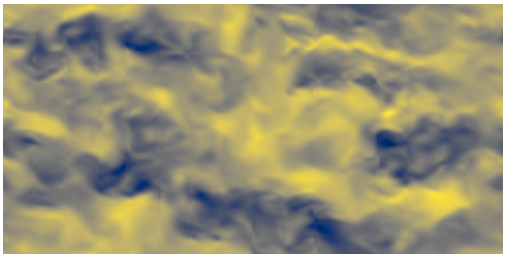
(a) projected $p = 1$, input



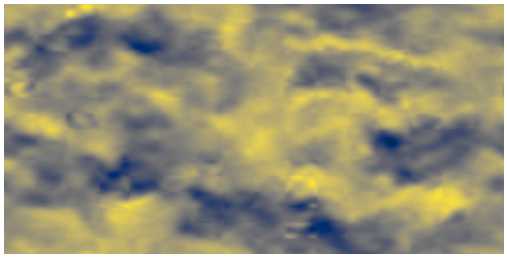
(b) projected $p = 3$



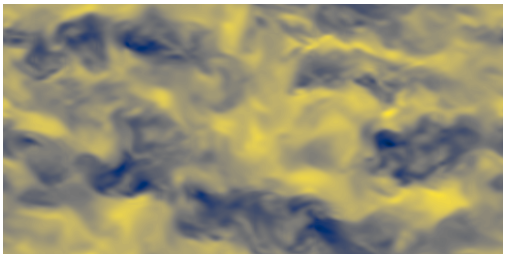
(c) incremental reconstruction 1



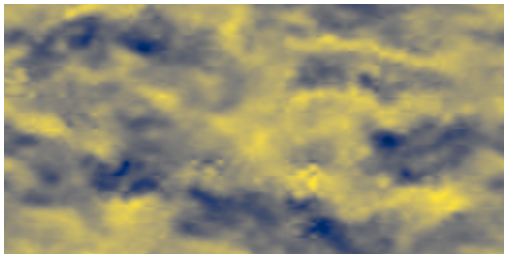
(d) projected $p = 5$



(e) incremental reconstruction 2



(f) projected $p = 7$



(g) incremental reconstruction 3

Figure 4.11: Qualitative comparison of incremental super-resolution against the true solution at each reconstruction step. The left column is the truth at each incremental reconstruction step, the right column is the reconstruction. Streamwise velocity contours at $y^+ \approx 247$.

frequency content, but they are clearly of a different character than the turbulence in the projected snapshots.

The deviation in the character of the oscillations from true projected turbulence may come down to the loss function. The loss function for all training in this paper is the mean squared error between actual and predicted basis function coefficients. Nothing in this loss encodes turbulent structures, potentially causing the spurious oscillations we see in the final super-resolved output. Also, since we use basis function coefficients as input and output, the results are dependent on the choice of basis.

4.2 Super-Resolution in Three Dimensions

In this section we will determine if super-resolution reconstruction continues to function in three dimensions. Since flow-field parameters increase exponentially with dimension, it is not immediately obvious that reconstruction performance will be retained. In fact, we expect performance to degrate, but for adaptation we simply need a method that is directionally correct in its reconstruction. We will propose a network architecture able to capture, at least approximately, three dimensional geometry variation. We will also explore the required model sizing under simple architecture, input, and training assumptions. Finally, as in the previous chapter, we will explore the performance of the resulting networks across flow conditions to determine their suitability for adaptation.

4.2.1 Methodology

We will begin by discussing out primary evaluation technique, which remains spectral comparison. For the sake of simplicity, we will generate our spectra on turbulent channels at various Reynolds numbers. The turbulent channel has two statistically homogeneous directions, making it a natural choice for the generation of spectra. Each turbulent energy spectrum is computed as a set of one dimensional discrete Fourier transforms on 200 sam-

pled points of the streamwise velocity mean deviation in the streamwise direction. While the super-resolution network makes predictions for all velocities, we will use accuracy in the streamwise direction as a proxy for overall network accuracy. For a fixed wall distance, the energy spectrum is sampled at 40 evenly spaced spanwise locations on each snapshot. The spanwise averaged spectra from 20 snapshots are then combined for the final energy spectrum. This process significantly reduces noise in the final result. The time averaging process also serves the dual purpose of minimizing out of plane effects. Since the reconstruction operates on 3D elements but the spectral sampling is only in 2D, low frequency energies may not match when taking only a single snapshot. However, this effect is eliminated by time averaging since flow properties along a plane will converge over time.

Training data is generated from four cases. The first three are turbulent channels at $Re_\tau = 395$, $Re_\tau = 590$, and $Re_\tau = 950$. Each is run at $p = 15$, this high order ensures the data can be projected down to multiple lower orders for network training. The $Re_\tau = 395$ case uses 768 elements, the $Re_\tau = 590$ case uses 1400 elements, and the $Re_\tau = 950$ case uses 1600 elements. Each mesh uses smaller elements near the channel walls, providing some cell Reynolds number variation in that direction. The final training set is a periodic hill with 1024 elements, this adds some geometry curvature to the training set. For each case and each network order, 50 snapshots are used to generate training data.

Testing data comes from three turbulent channel simulations at the same Reynolds numbers as the training data: $Re_\tau = 395$, $Re_\tau = 590$, and $Re_\tau = 950$. The meshes used are also the same, but the case is different. This time, a $p = 7$ case is used. This ensures the test data is not part of the training set. As with the 1D case, we ensure testing occurs on the same meshes for reconstruction accuracy, this will continue through to adaptation.

The training technique is kept consistent to ensure a fair comparison between the various networks. Each network is trained with batched gradient descent with a batch size of 256. The Adam optimizer is used [74] with a learning rate of 10^{-3}. Training uses early stopping,

proceeding until loss on the validation set stagnates or rises. The number of training samples varies significantly depending on data set selection, ranging from 30,720 when only the $Re_\tau = 395$ set is used, to 191,680 when all data sets are employed.

4.2.2 Network Design

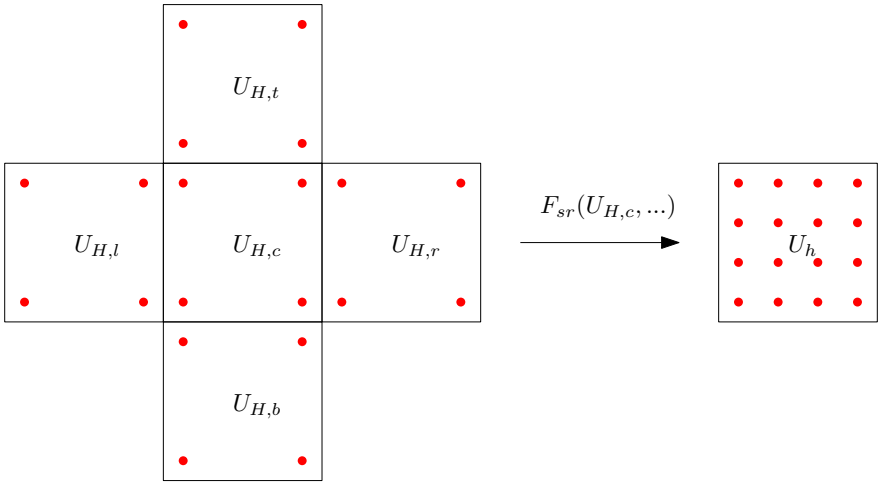


Figure 4.12: The network takes state from a coarse space and approximates the solution in a fine space.

The super-resolution network computes a state correction one element at a time. For each element the state on a subset of neighboring elements is also considered for input. In this work this subset is the set of elements directly across a face from the element of interest. This comes to a total of seven elements for the three dimensional networks. The super-resolution neural-network input consists of normalized basis function coefficients with auxiliary scaling and rotation information. Only normalized momentum basis function coefficients are input to the network. Since the flow is nearly incompressible, the density is nearly constant and the flow momentum effectively reduces to velocity.

As in previous network setups, we draw heavily from Pradhan and Duraisamy's [99] variational multiscale super-resolution network. In order to generalize the network across flow conditions, the basis function coefficients are normalized by removing the mean velocity value from each component and dividing out the root mean square variation of the central

element. The mean and root mean square normalizing values are defined as

$$\mathbf{u}_{m,e} = \frac{\int_{\Omega_e} \mathbf{u}_e \, \mathrm{d}\Omega}{|\Omega_e|} \qquad \mathbf{u}_{\mathrm{rms},e} = \sqrt{\frac{\int_{\Omega_e} \left(\mathbf{u}_e - \mathbf{u}_{m,e}\right)^2 \, \mathrm{d}\Omega}{|\Omega_e|}}, \tag{4.6}$$

where $\mathbf{\Omega}_e$ is the domain of element e, and $\mathbf{u}_e$ is the continuous velocity vector field over element e.

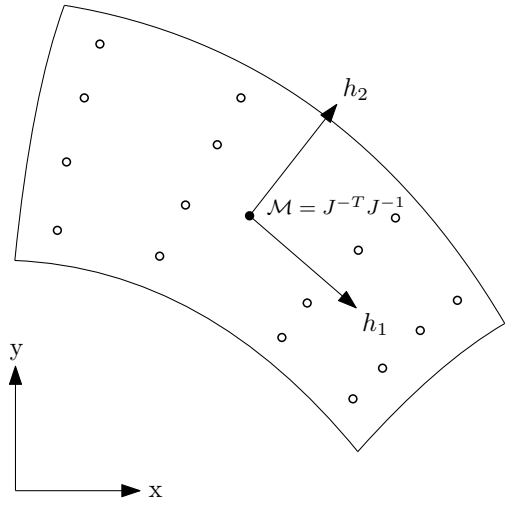


Figure 4.13: Complex geometry curvature information is reduced to a single tensor in this simple model.

The approximate size and orientation of an element is measured by computing the mesh implied metric tensor [124] at a single arbitrary point within the element. The metric tensor provides a yard-stick for measuring distance over a field. This serves our purposes because we can learn how an element is stretched and warped by computing the mesh implied metric tensor. Distance under the metric $l_{\mathcal{M}}$ between two points a and b is defined as

$$l_{\mathcal{M}}\left(\overrightarrow{ab}\right) = \int_0^1 \sqrt{\overrightarrow{ab}^T \mathcal{M}\left(a + \overrightarrow{ab}s\right)\overrightarrow{ab}}\, ds. \tag{4.7}$$

The mesh implied metric tensor is that which would have $l_{\mathcal{M}} = 1$. This tensor is computed by using information from the reference space to global space transformation on that element. In finite-elements we use a reference space, $\vec{\xi}$, for integration that is common between

all elements. Physical space, $\vec{x}$, elements are defined relative to this reference space by a transformation function $\vec{x} = \mathcal{G}\left(\vec{\xi}\right)$. The derivative of this transformation $J = \partial\vec{x}/\partial\vec{\xi}$ is a Jacobian matrix useful for integration and differentiation. In a super-resolution context we use the Jacobian matrix to compute the mesh implied tensor using a relationship from Yano [124]

$$\mathcal{M} = J^{-T}J^{-1}. \tag{4.8}$$

The final network uses the information from the metric tensor in two ways. The eigenvectors of the metric tensor are principal stretching directions. A diagram of these directions is shown in Figure 4.13. The root mean square vector u_{rms} is rotated to align with the principal stretching directions and used to compute a cell Reynolds number roughly aligned with the element. The logarithm of this value is used to keep input scaling order one. We also include rotational information from the off diagonal components of the logarithm of the metric tensor. The final calculation for the reconstructed state looks like

$$\begin{aligned}&\mathbf{U}_{h,c} = \\ &F_{sr}\left(\frac{\mathbf{U}_{H,c,d} - u_{m,d}}{u_{\text{rms},d}}, \frac{\mathbf{U}_{H,n,d} - u_{m,d}}{u_{\text{rms},d}}, \log_{10}\frac{h_d u_{\text{rms},d}}{\nu}, \log_{10}\mathcal{M}_{r\neq c}\right) u_{\text{rms},d} + \mathbf{U}^H_{h,c} \forall d \in \mathcal{D}, n \in \mathcal{N},\end{aligned} \tag{4.9}$$

where the basis function coefficients on the fine space for the central element $\mathbf{U}_{h,c}$ are reconstructed by addition of the network F_{sr} output to the central element's prolonged coarse space state $\mathbf{U}^H_{h,c}$. $\mathcal{D}$ is the set of ranks used in the network, one for each direction in this case. $\mathcal{N}$ is the set of all neighboring elements, this set consists of six elements, one across each face. $\mathbf{U}_{H,c,d}$ denotes the full state vector $\mathbf{U}$ restricted to the central element c and rank d. Likewise $\mathbf{U}_{H,n,d}$ denotes the full state vector restricted to neighbor n and rank d.

Thus far, element neighbors have only spanned periodic boundaries so no boundary handling was required. In three dimensions we will encounter walls across which there is no other state. We assume all walls are no slip and simply set the full neighboring state in the network input

to zero during training and testing.

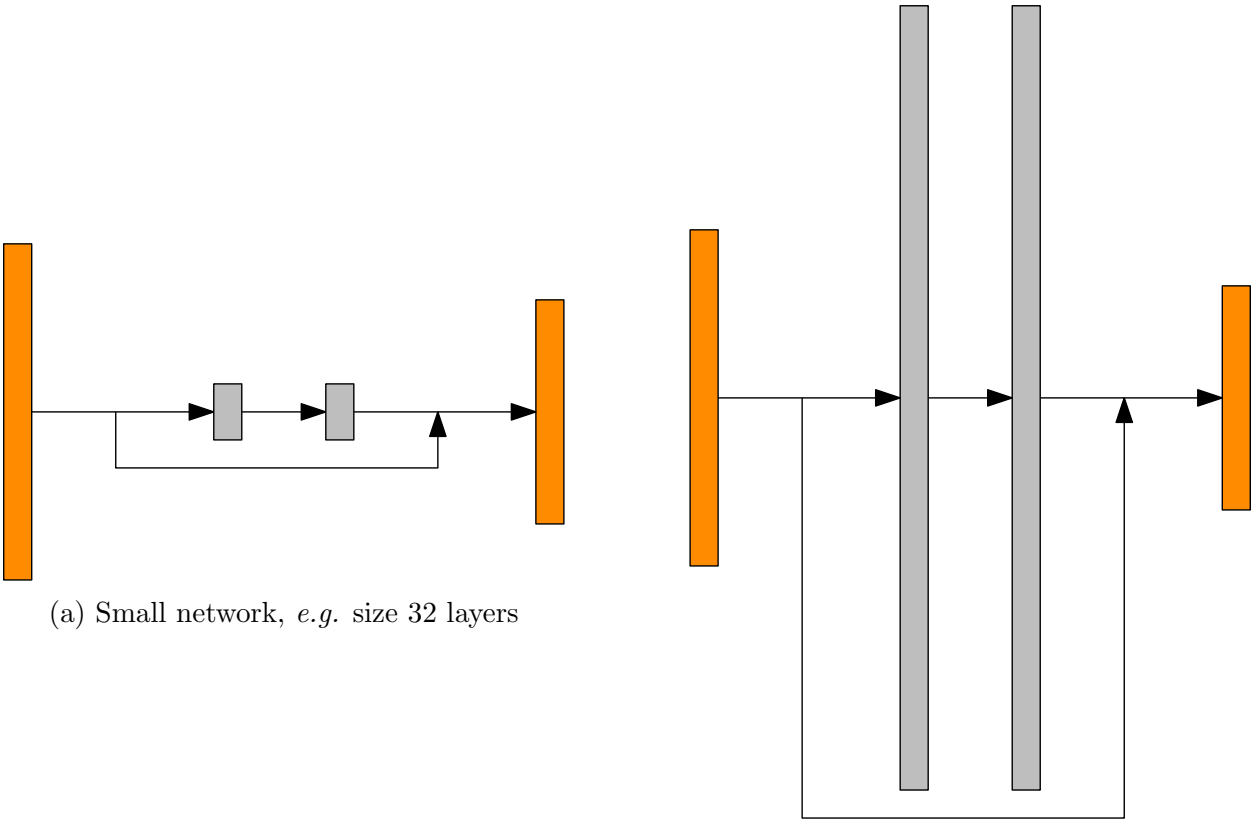
4.2.3 Network Sizing Study

For the present tests of super-resolution reconstruction in three dimensions the network architecture is a simple fully connected network. Fully connected networks should be sufficient to model any function, including our super-resolution problem. However, one should expect a relatively poorly conditioned loss landscape from fully connected networks. We will see some of this difficulty when testing for appropriate network size. In this study, each network will have two hidden layers, Figure 4.14 shows the architecture. This is to increase the number of connections in the network relative to a single hidden layer, while not adding too much depth for this simple architecture. We will vary the hidden layer size by orders of magnitude to roughly find an appropriate network size.

For training and testing data we restrict our attention to the $Re_\tau = 395$ turbulent channel. Training data is run at $p = 15$ and projected to the test orders $p = 3$ and $p = 5$. Testing data is from a separate simulation run at $p = 7$ on the same computational mesh. Training of each network proceeds as described in Section 4.2.1.

Testing focuses on four hidden layer sizes chosen to span a range of magnitudes: 32, 128, 512, and 2048. This leads to models with 65,672, 273,032, 1,286,792, 8,290,952 parameters respectively. The training data set is identical for each case consisting of 30,720 samples. 7,680 validation samples are tested during training.

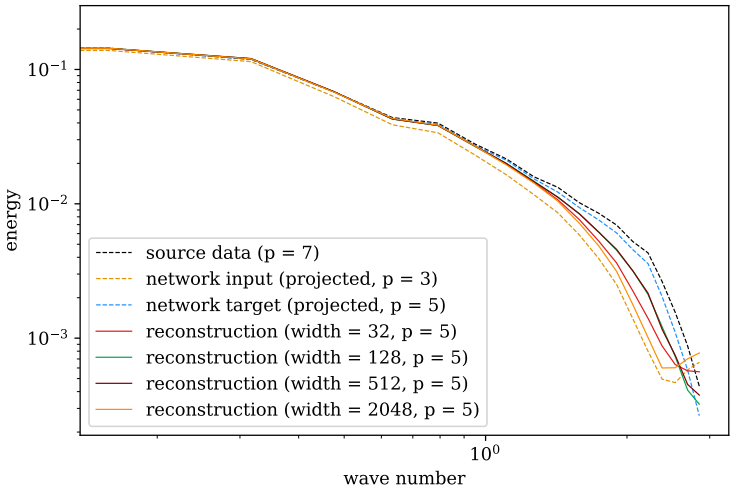
Spectral reconstruction results are shown in Figure 4.15. The story remains remarkably consistent at the tested wall distances. It appears the network with size 32 hidden layers performs quite poorly. This can be explained by the extremely restricted predictive power of the network. The next two networks with size 128 and 512 hidden layers perform nearly identically at all stations. Performance once again degrades in the case of the largest network. This performance degradation can be entirely explained by the inability to train the network.



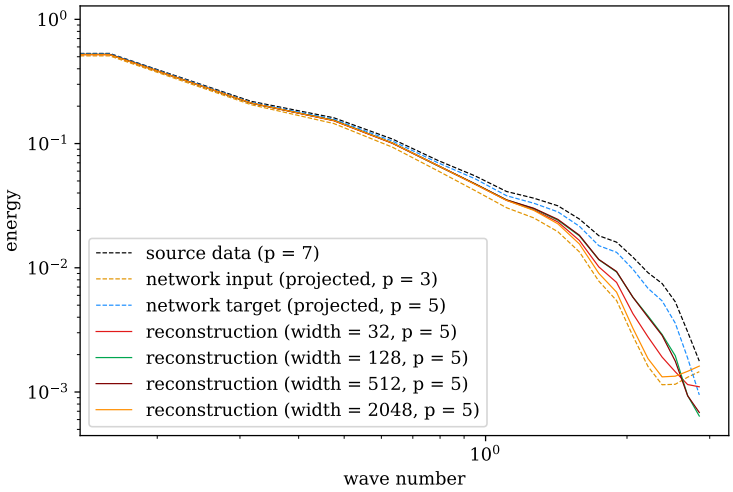
(a) Small network, *e.g.* size 32 layers

(b) Large network, *e.g.* size 2048 layers

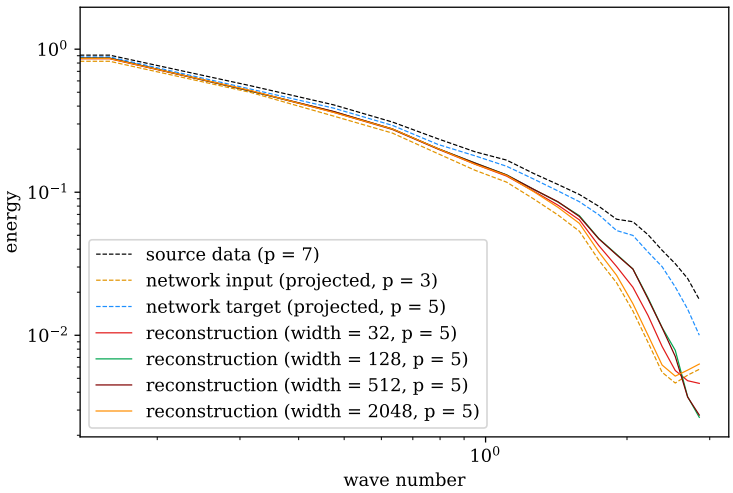
Figure 4.14: Example network architectures for sizing study. Each network is fully connected with two hidden layers. Input and output layers are in orange, hidden layers are in grey. Hidden layer size varies by orders of magnitude.



(a) $y^+ \approx 356$



(b) $y^+ \approx 198$



(c) $y^+ \approx 40$

Figure 4.15: Spectra for network sizing study at various wall distances. All data are collected on $Re_\tau = 395$ turbulent channel data.

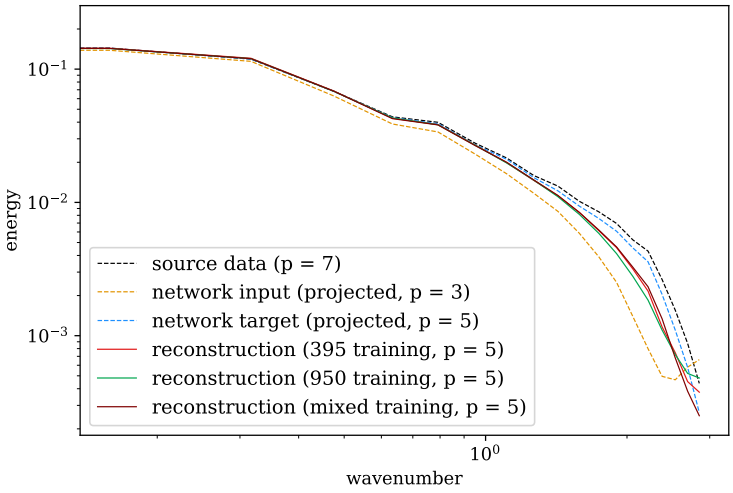
Despite using batched gradient descent, training loss stalls very quickly and drops little relative to the other networks. This story holds as expected with the validation loss. Perhaps the loss landscape of the large network is sufficiently poor that the present training technique fails. This failure indicates much better reconstruction performance may be possible with differences in network architecture and training.

For now, we will accept the current reconstruction quality and proceed with further testing. The choice now is between the size 128 and 512 networks. Going forward we will use the larger size 512 network for testing and adaptation. This decision is justified by a couple observations. First, we have kept the training set for this test relatively small at around 30,000 samples, this will grow when considering more data sets and a larger network should have more capacity to model this size increase. Second, we are already employing early stopping to prevent overfitting, so a larger network should not present an over fitting difficulty.

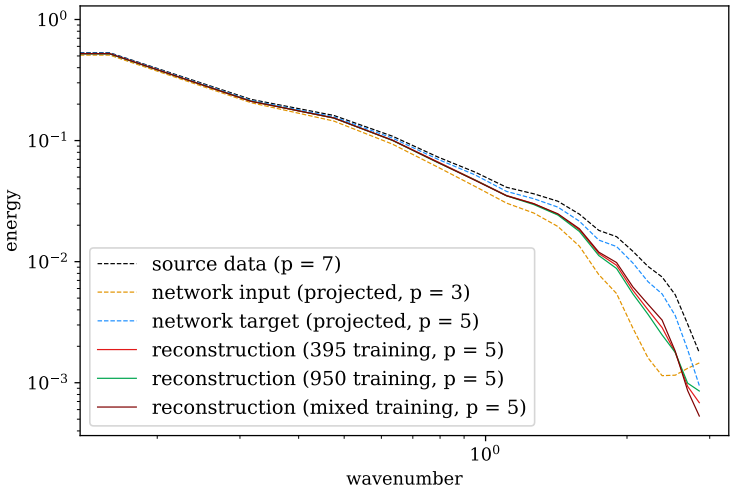
4.2.4 Influence of Training set and Reynolds Number

Similar to the first chapter, we will explore reconstruction quality across various Reynolds numbers for various training sets. In the 1D setting we observed excellent reconstruction performance, but poor extrapolation across Reynolds numbers. We also observed relatively poor translation of a network trained on a single viscosity to data at different viscosities. The purpose of running these tests in 3D is to determine the appropriate architecture for adaptation. Ideally it would be best to use a single network across data sets, but this may prove infeasible if reconstruction performance significantly breaks down. In that situation it may be best to keep flow conditions consistent for each network and use a different network for each adapted case.

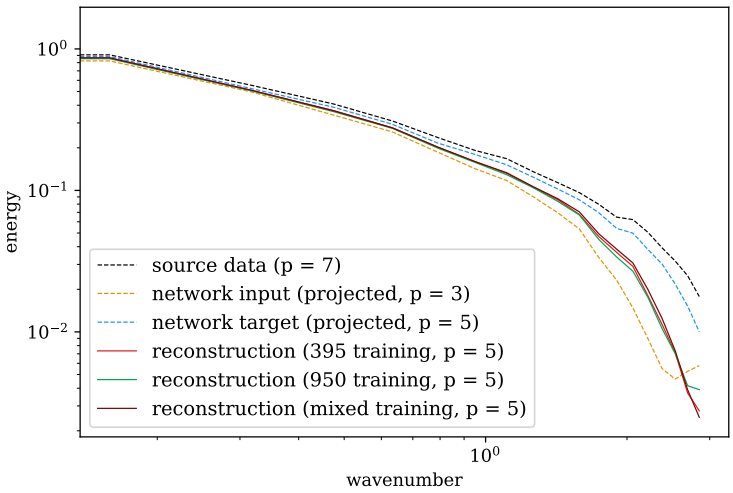
For the network parameterization we will use the results of the network sizing study. In that study we observed a network with two hidden layers of size 512 to be both reasonable and somewhat over sized for reconstruction from $p = 3$ to $p = 5$. We will continue to test



(a) $y^+ \approx 356$

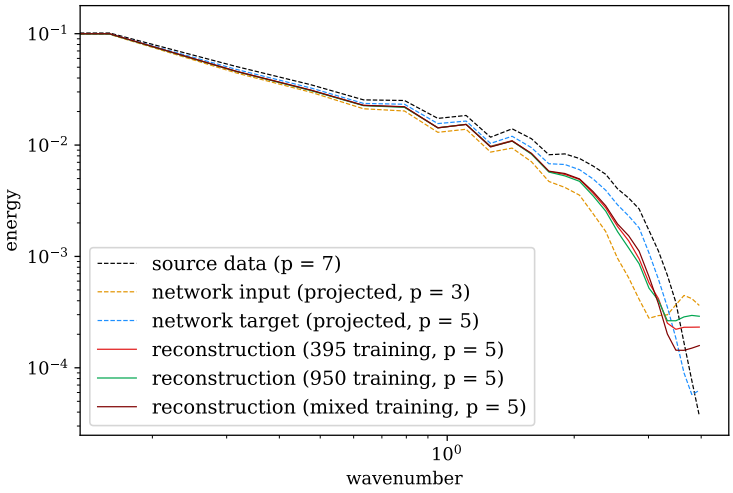


(b) $y^+ \approx 198$

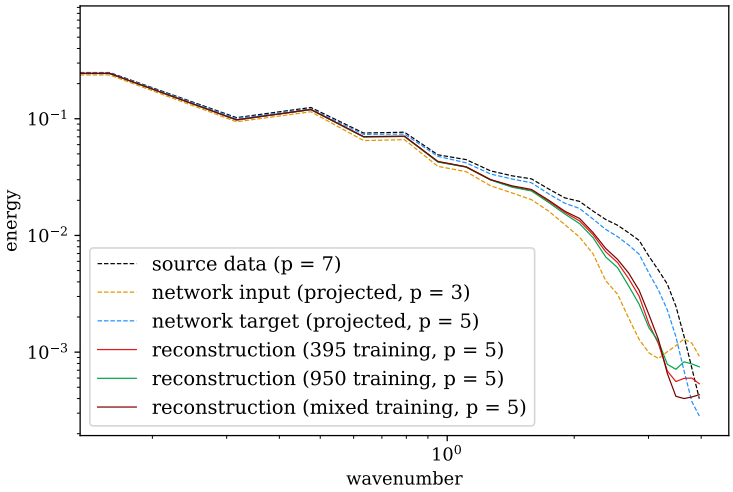


(c) $y^+ \approx 40$

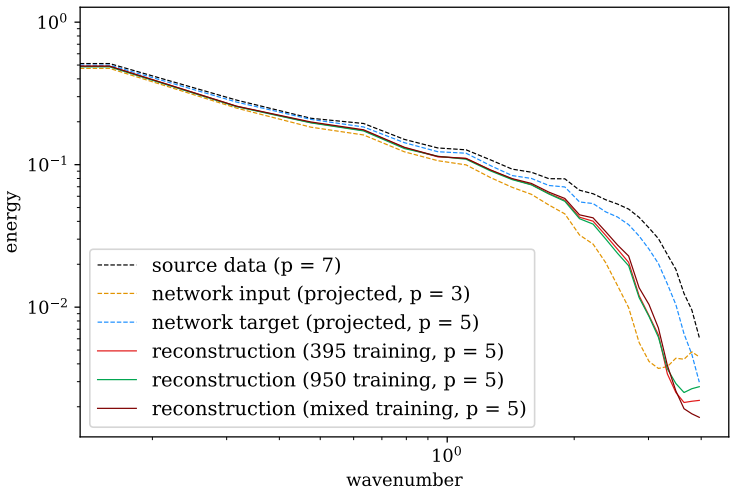
Figure 4.16: Streamwise turbulent energy spectrum reconstruction comparisons for a turbulent channel data set at $Re_\tau = 395$.



(a) $y^+ \approx 855$



(b) $y^+ \approx 475$



(c) $y^+ \approx 95$

Figure 4.17: Streamwise turbulent energy spectrum reconstruction comparisons for a turbulent channel data set at $Re_\tau = 950$.

reconstruction between $p = 3$ and $p = 5$, but this time we will include more data and use the fact that the network is over sized to account for the additional data. Training set sizes are held fixed for each Reynolds number and training data is only altered one set at a time for simplicity.

As usual, we will explore two performance measures: spectral and qualitative comparison. Spectrum will once again use streamwise velocity reconstruction quality as a proxy for overall network performance. Qualitative comparison of reconstructed snapshots is more difficult in 3D since the required reconstruction deltas have proved relatively small, but the comparisons are nonetheless provided for completeness.

Spectral comparisons are shown in Figures 4.16 and 4.17. Each figure shows network input, target output, and reconstructed profiles. Each network performing reconstruction is trained on a different set of data. These sets are $Re_\tau = 395$ data only, $Re_\tau = 950$ data only, and mixed $Re_\tau = 395$, $Re_\tau = 590$, $Re_\tau = 950$, and periodic hills data. It is important to note the number of training samples does increase for each of these data sets. The number of training samples is 30,720, 64,000, and 191,680 respectively. This should give an advantage to the fully mixed data set, but it is not yet clear that the additional data will sharpen or degrade the network.

In both figures and for all reconstructions, performance significantly degrades approaching the wall. This is expected, since the flow-field is significantly more complex near the wall. The spread between each reconstructed data set is also very small for each test case, this may be for several reasons. Baseline reconstruction quality is significantly worse than the 1D case. Since all reconstructions are reasonably distant from the target state, the small differences between them matter less than if they were nearly exact. It is also possible the constant network size is hampering performance, especially of the fully mixed data set. This could indicate that the oversizing of the network is insufficient to handle the increased training set size.

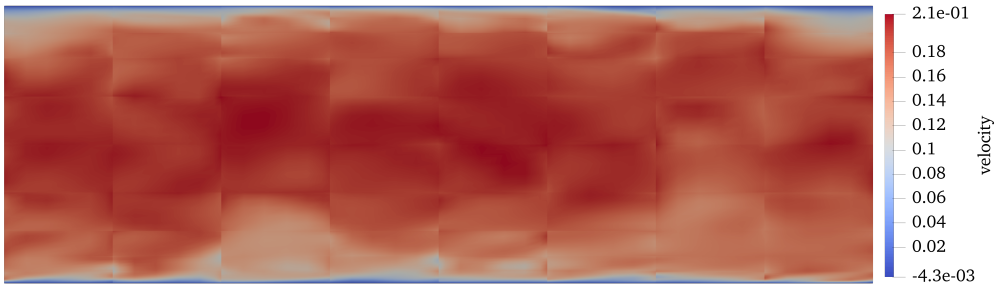
It appears that reconstruction quality for the network trained with exclusively $Re_\tau = 950$ data is consistently worse than the other networks. This difference holds even on the $Re_\tau = 950$ test set which is not expected. The difference is not large but consistent. This may be due to the relatively high-frequency state of that data set combined with relatively few samples, at least compared to the fully mixed data set. Likewise the fully mixed data set appears to perform slightly better than the others across the board. This could indicate that the network, still at fixed size, generalizes quite well with increased data. The good performance of the $Re_\tau = 395$ trained network on the $Re_\tau = 950$ test set is an encouraging result for network generalization. If that generalization holds for training at different Reynolds numbers, it would explain some of the performance of the fully mixed network.

Qualitative comparisons are shown in Figures 4.18, 4.20, 4.20, and 4.21. While differences are relatively difficult to see, they are included for completeness. Increased frequencies are observable in the reconstructed (middle) states while the overall character of the coarse solution is retained as expected. It also appears that inter-element discontinuities decrease for the reconstructed states. This is visual corroboration for the reduced high frequencies seen for the reconstructed profiles in the spectral plots.

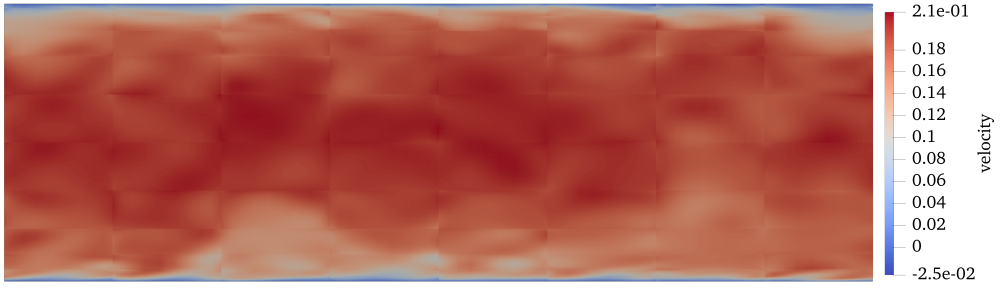
4.3 Summary

We begun this chapter by testing super-resolution reconstruction in two dimensions. High order turbulent channel data was sliced along wall normal planes, and projected to 2D DG state. While initial reconstruction quality was good, it could be improved by performing reconstruction incrementally. It is shown that better reconstruction results can be achieved performing a series of small reconstructions that a single large one. This observation reinforces the idea that improved network architectures can significantly improve reconstruction quality.

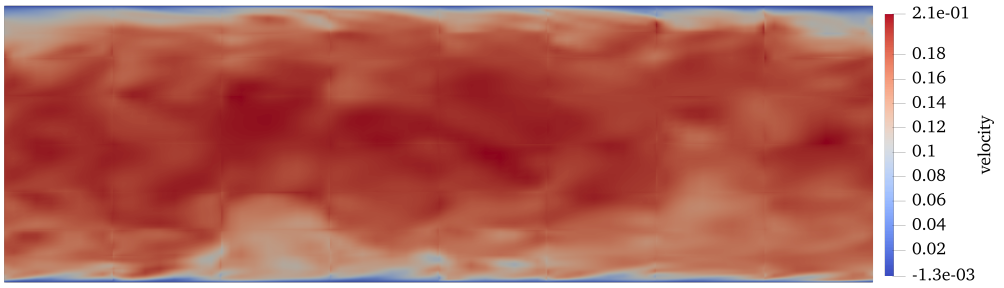
We then moved on to three dimensional reconstruction. A simple fully connected network



(a) input state, $p = 3$

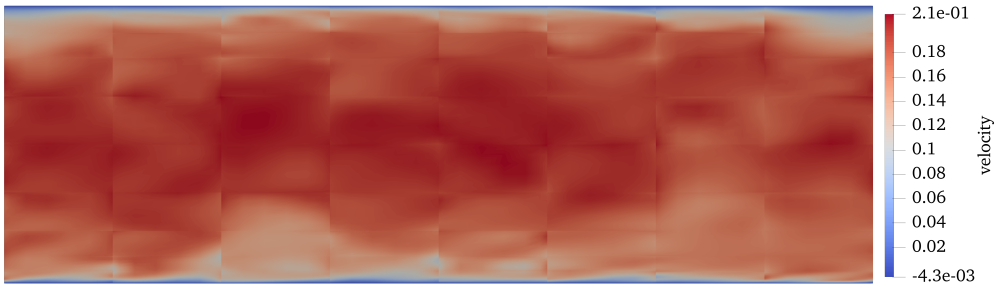


(b) reconstructed state, $p = 5$

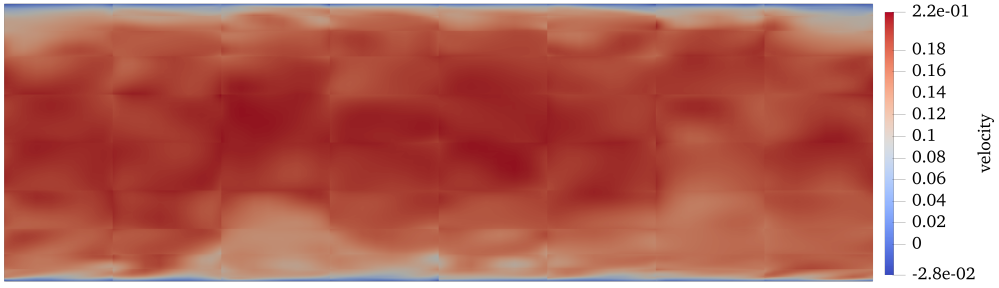


(c) target state, $p = 5$

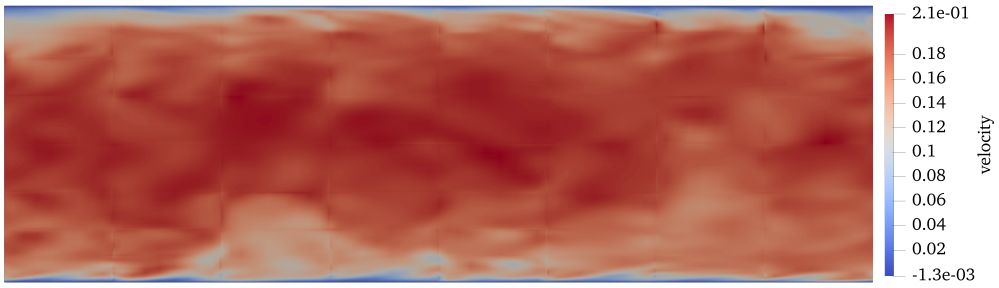
Figure 4.18: Selected snapshot reconstruction where a network trained on $Re_\tau = 395$ turbulent channel flow data is used to reconstruct an $Re_\tau = 395$ turbulent channel flow-field.



(a) input state, $p = 3$

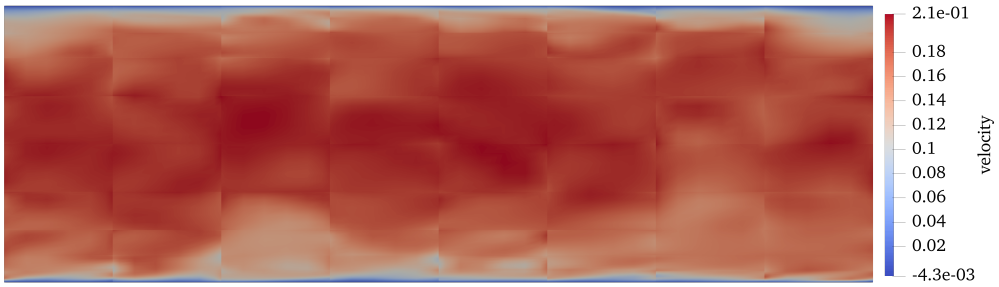


(b) reconstructed state, $p = 5$

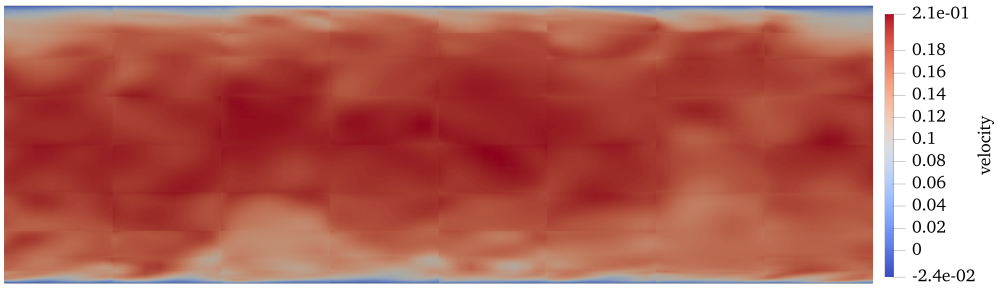


(c) target state, $p = 5$

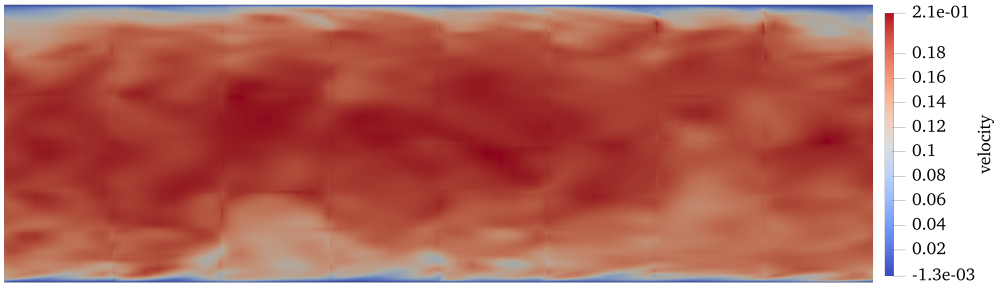
Figure 4.19: Selected snapshot reconstruction where a network trained on $Re_\tau = 950$ turbulent channel flow data is used to reconstruct an $Re_\tau = 395$ turbulent channel flow-field.



(a) input state, $p = 3$

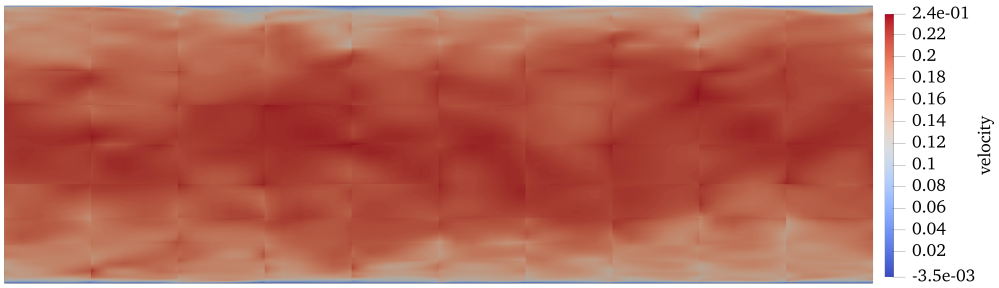


(b) reconstructed state, $p = 5$

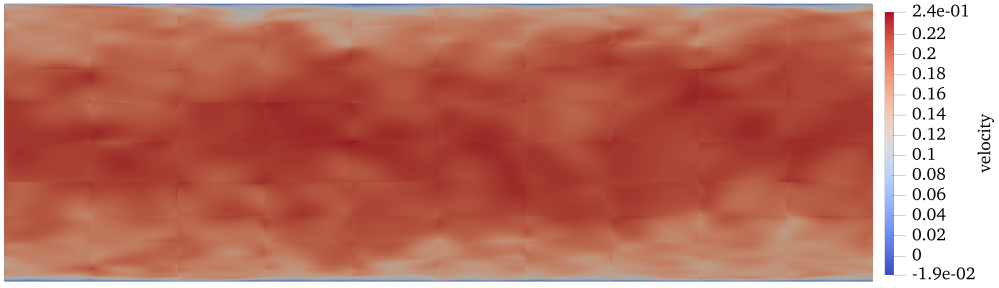


(c) target state, $p = 5$

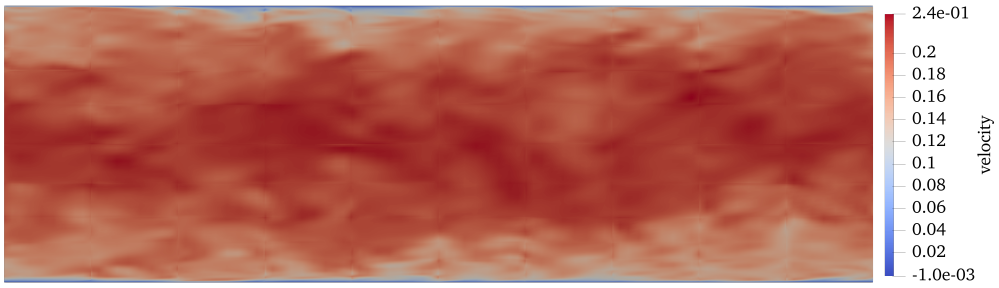
Figure 4.20: Selected snapshot reconstruction where a network trained on mixed turbulent channel and periodic hill data is used to reconstruct an $Re_\tau = 395$ turbulent channel flow-field.



(a) input state, $p = 3$

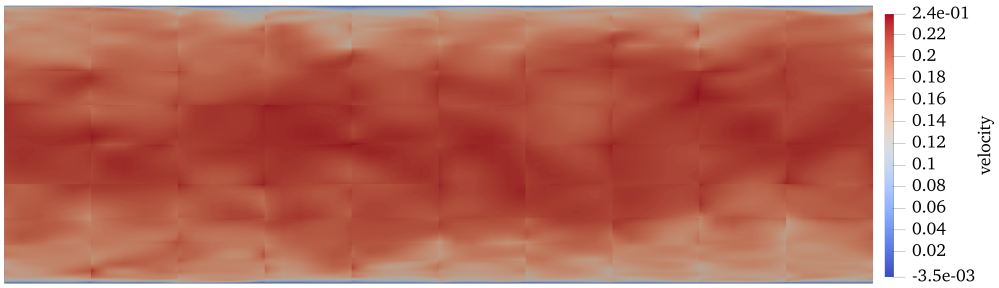


(b) reconstructed state, $p = 5$

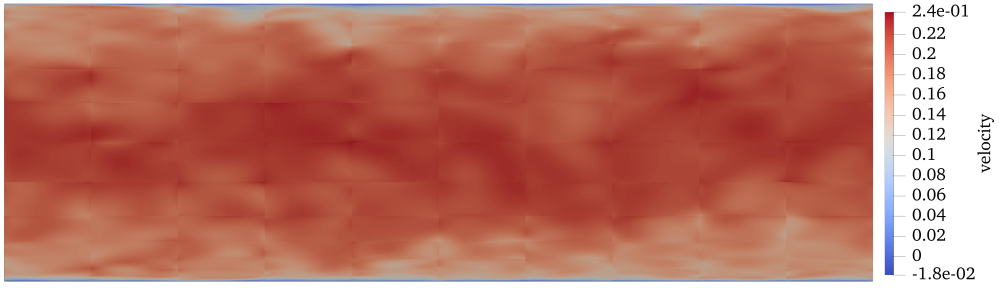


(c) target state, $p = 5$

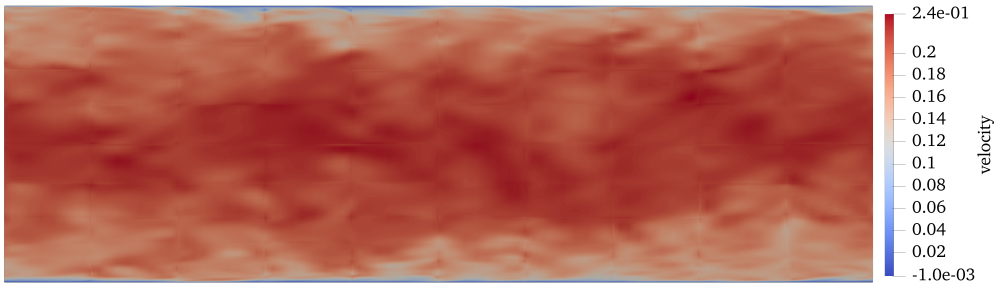
Figure 4.21: Selected snapshot reconstruction where a network trained on $Re_\tau = 395$ turbulent channel flow data is used to reconstruct an $Re_\tau = 950$ turbulent channel flow-field.



(a) input state, $p = 3$

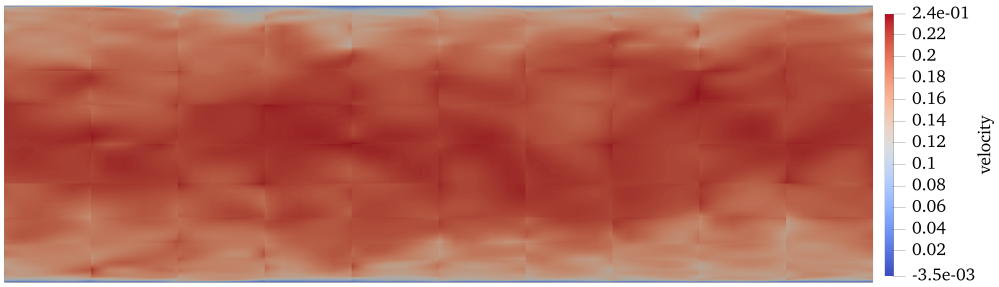


(b) reconstructed state, $p = 5$

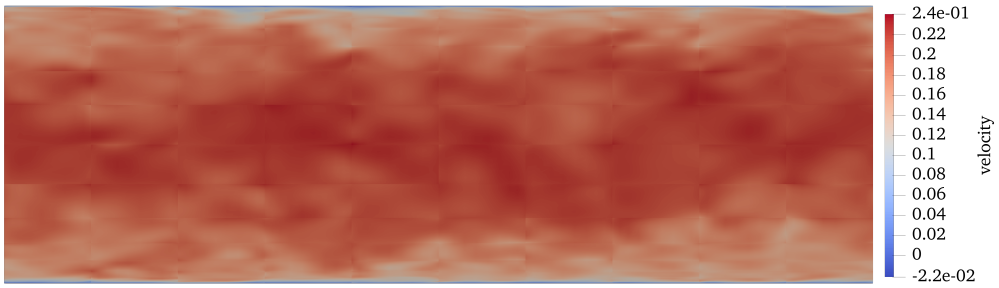


(c) target state, $p = 5$

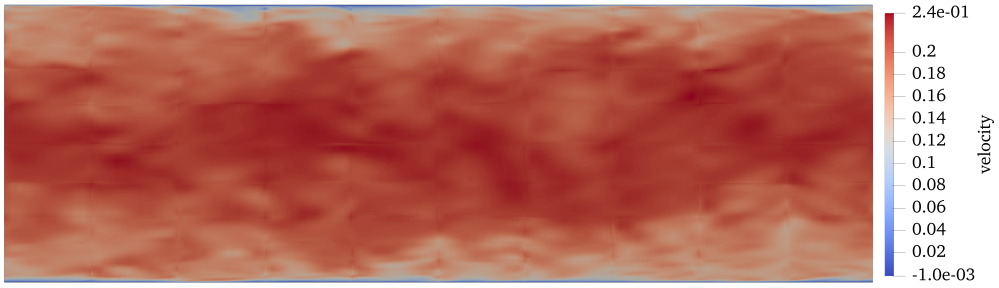
Figure 4.22: Selected snapshot reconstruction where a network trained on $Re_\tau = 950$ turbulent channel flow data is used to reconstruct an $Re_\tau = 950$ turbulent channel flow-field.



(a) input state, $p = 3$



(b) reconstructed state, $p = 5$



(c) target state, $p = 5$

Figure 4.23: Selected snapshot reconstruction where a network trained on mixed turbulent channel and periodic hill data is used to reconstruct an $Re_\tau = 950$ turbulent channel flow-field.

architecture was used for this testing. Auxiliary information about element orientation was introduced to the network, now that elements can be arbitrarily rotated in three dimensions. This network architecture was tested at various sizes leading to a selection for further testing. Networks were then trained and tested on various sets of data to determine network generalizability. It was concluded that a single network for a variety of cases is appropriate as long as it has been exposed to similar data in training.

Chapter 5

Super-Resolution Adaptation in 3D

5.1 Introduction

In this chapter, two error indicators are introduced. One simply measures the predicted state correction by the super-resolution network. The second uses super-resolution reconstruction in an adjoint weighted residual setting. Both indicators are tested on a contrived channel problem where all elements are identically shaped. The state correction indicator is then tested against another error indicator that uses the mean velocity gradient with the same refinement strategy. Both super-resolution-based indicators are then tested on a periodic hill geometry. A slightly modified version of the state difference indicator is then tested on the trailing edge cooling slot problem presented in [54].

Parts of this chapter appear in or are adapted from our previously published paper [86].

5.2 Super-Resolution-Based Error Indicators

5.2.1 State Difference

We have a super-resolution model that predicts a normalized correction to the input state in terms of basis function coefficients. A first temptation is to simply use the magnitude of the output as the error indicator. This would be a reasonable choice were it not for the fact that the magnitude of the error indicator would depend on a discrete norm. Each order will have a different number of basis function coefficients, making the value of this indicator basis dependent. Specifically, it would tend to over estimate the error at higher orders, which is contrary to expectations. To remove the basis dependence, we can instead use an indicator based on the state represented by the basis function coefficients, instead of the coefficients themselves.

The super-resolution model predicts the correction required to the velocity field within an element. To change the state of the full element, the density is held constant, and internal energy is set such that pressure is held constant. With this information we are able to compute the super-resolved set of basis function coefficients, $\mathbf{U}_{sr}$. We are then able to compute the continuous super-resolved state by linear combination with the set of basis functions on each element

$$\mathbf{u}_{h,sr} = \sum_{i=1}^{n} \left(\left(\mathbf{U}_{sr,i} + \mathbf{U}_{h,i}^{H} \right) * u_{rms} \right) \phi_{h,i}, \tag{5.1}$$

where $\mathbf{U}_{h,i}^{H}$ is the original element state injected into the fine approximation space, and $\phi_{h,i}$ is the i^{th} fine basis function coefficient for the element. With this super-resolved state in mind, the state difference error indicator on element k is

$$\mathbf{e}_{k,s} = \int_{\Omega_k} \left(\mathbf{u}_{h,sr,k,s} - \mathbf{u}_{h,k,s}^{H} \right)^2 \mathrm{d}\Omega \qquad e_k = \sum_{s} \mathbf{e}_{k,s}, \tag{5.2}$$

where the final indicator is the sum of indicators for each rank s. This equation is simply the two-norm of the difference between the reconstructed and original states for each state rank. The error indicator is at that point a vector, each entry of this vector is summed to find the final error indicator. No absolute value is necessary since all outputs of the two-norm are non-negative.

5.2.2 Entropy-Adjoint-Weighted-Residual

For the derivation of the adjoint weighted residual error indicator, we follow Fidkowski and Darmofal [45]. For any discrete system $\mathbf{R}_h(\mathbf{U}_h) = \mathbf{0}$ with scalar output J, we can define an adjoint vector $\boldsymbol{\psi}$ as the sensitivity of J with respect to residual perturbations

$$\delta J_h \equiv J_h(\mathbf{U}_h + \delta\mathbf{U}_h) - J_h(\mathbf{U}_h) \equiv \boldsymbol{\psi}_h^T \delta\mathbf{R}_h \tag{5.3}$$

where $\delta\mathbf{u}_h$ satisfies

$$\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\delta\mathbf{U}_h + \delta\mathbf{R}_h = \mathbf{0}. \tag{5.4}$$

If the equation set and outputs are differentiable we have

$$\delta J_h = \frac{\partial J_h}{\partial \mathbf{U}_h}\delta\mathbf{U}_h = \boldsymbol{\psi}_h^T \delta\mathbf{R}_h = -\boldsymbol{\psi}_h^T \frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\delta\mathbf{U}_h, \tag{5.5}$$

rearranging yields the adjoint equation

$$\left(\frac{\partial \mathbf{R}_h}{\partial \mathbf{U}_h}\right)^T \boldsymbol{\psi}_h + \left(\frac{\partial J_h}{\partial \mathbf{U}_h}\right)^T = 0. \tag{5.6}$$

Returning to the adjoint definition

$$\delta J_h \equiv \boldsymbol{\psi}_h^T \delta\mathbf{R}_h, \tag{5.7}$$

we can replace the δ quantites with a difference of two discretization levels to find an error indicator. Assuming we are working from a solved state removes one residual entry resulting in the error estimate

$$J_h\left(\mathbf{U}_h^H\right) - J_h\left(\mathbf{U}_h\right) = -\boldsymbol{\psi}_h^T \mathbf{R}_h\left(\mathbf{U}_h^H\right). \tag{5.8}$$

If we define the adjoint perturbation as $\delta\boldsymbol{\psi} \equiv \boldsymbol{\psi}_h^H - \boldsymbol{\psi}_h$ we can rearrange Equation 5.8 as

$$\delta J \approx -\left(\boldsymbol{\psi}_h^H\right)^T \mathbf{R}_h\left(\mathbf{U}_h^H\right) + \left(\delta\boldsymbol{\psi}_h\right)^T \mathbf{R}_h\left(\mathbf{U}_h^H\right). \tag{5.9}$$

The first term is zero for discontinuous Galerkin, so the final form of the error estimate we will use with entropy variables is

$$\delta J \approx \left(\delta\boldsymbol{\psi}_h\right)^T \mathbf{R}_h\left(\mathbf{U}_h^H\right). \tag{5.10}$$

5.2.3 The Entropy Adjoint

The Navier–Stokes equations admit an entropy function for which the corresponding entropy variables symmetrize both the inviscid and viscous terms [66]. This entropy function can be uniquely defined (up to additive and multiplicative constants) as

$$U = -\rho S/R, \quad S = c_v \ln p - c_p \ln \rho, \tag{5.11}$$

where p is the pressure (not to be confused with the order; the meaning will be clear from the context), ρ is the density, and c_v and c_p denote the specific heat capacities at constant volume and pressure, respectively. Differentiating with respect to the conservative state $\mathbf{u} = [\rho, \rho\vec{V}, \rho E]^T$ yields the entropy variables,

$$\mathbf{v} = U_{\mathbf{u}}^T = \left[\frac{\gamma}{\gamma-1} - \frac{s}{R} - \frac{1}{2}\frac{\rho\|\vec{V}\|^2}{p}, \frac{\rho\vec{V}}{p}, -\frac{\rho}{p}\right]^T. \tag{5.12}$$

The entropy function is conserved in the computational domain, with the corresponding entropy flux defined as $\vec{F}_s(U) = \vec{V}U = -s\rho\vec{V}/R$.

For a specific choice of the entropy function, the corresponding entropy variables symmetrize the governing equation and therefore satisfy the unsteady adjoint equation for one specific output [47],

$$J = \int_{\partial\Omega} \vec{F}_s \cdot \vec{n} dS - \int_{\Omega} \mathbf{v}^T \nabla \cdot (\mathbf{K}\nabla\mathbf{u}) d\Omega. \tag{5.13}$$

This output is an entropy balance statement for the computational domain: the first term represents the net outflow of the entropy through the boundary, while the second term denotes the total generation (dissipation) of the entropy inside the domain. Since the output J originates from integrating the adjoint equation, an integral form for the conservation of entropy, it measures the total entropy rate of change in time. For steady-state systems, it should be strictly balanced, $J = 0$; while for unsteady systems, J should be directly tied to the physical entropy changes. However, in a discrete sense, J often suffers from discretization errors, which cause spurious entropy generation in the computational domain. Hence, adapting on J targets areas where the physical entropy production is not correctly approximated.

There are several advantages of the entropy-adjoint over the standard output adjoint. First of all, the entropy-adjoint is a function of the primal state, which is readily available without solving the adjoint equations. This is especially efficient for unsteady systems, as the backward time integration is not required for the unsteady entropy-adjoint. Furthermore, the entropy-adjoint is well-conditioned and stable as long as the state solution is stable, making it applicable to turbulent flow simulations.

We use an adjoint error indicator with the same form as Equation 5.10. The output adjoint ϕ is replaced by the entropy-adjoint v. The fine space adjoint component is computed using a super-resolution neural network. The reconstructed state is simply transformed directly into entropy variables for use in the adjoint. Our solver for these cases, *eddy*, uses

discontinuous Galerkin for temporal evolution in addition to spatial resolution. The solver is structured in time such that states take the form of four dimensional time slabs, with three spatial dimensions and one temporal dimension. Our super-resolution networks are trained to reconstruct single snapshots one element at a time. For this indicator, we reconstruct a full time slab by reconstructing the state at each time Lagrange node individually. We also use the unsteady residual for the temporal evolution of that time slab $\mathbf{R}'_h$. The final indicator is

$$e_k = \left| \left(\mathbf{v}_h \left(\mathbf{U}_{sr,h,k} \right) - \mathbf{v}_h \left(\mathbf{U}^H_{h,k} \right) \right)^T \mathbf{R}'_h(\mathbf{U}^H_{h,k}) \right|. \tag{5.14}$$

The absolute value is taken to ensure a positive indicator for the time slab, otherwise cancellation is allowed.

5.3 Adaptation Strategy

In the context of ILES, as no explicit sub-grid scale model is employed, the discretization error and the modeling error are tightly coupled. Refining the mesh, which reduces the discretization error and hence the modeling error, would yield asymptotically a DNS solution. However, for effective LES modeling, only scales that are large enough to affect our quantities of interest, *e.g.*, mean drag values, need to be well-resolved, while the smaller scales should be modeled. Therefore, effective adaptive LES should be targeting areas that are most important for accurate output predictions.

Since we are using a finite-element method, each element represents the solution with basis polynomials of a specified order. It is not necessary to hold this polynomial order constant across the computational domain. A sufficiently general code will allow the polynomial order to vary across elements, this can be used as a form of mesh adaptation. While increasing the polynomial approximation order for high-error elements will decrease error overall, this method is not perfect. Adapting order causes discontinuous jumps in the discretization resolution leading to artifacts in the final flow-field. Nevertheless, it is a simple and easy to

use adaptation technique enabled by high-order elements.

Our error indicators are set up to approximate the error on each element of a turbulent flow-field snapshot or time slab. In theory, we could adapt by increasing the order of the worst elements after sampling only a single snapshot. However, practice indicates the error indicators are quite noisy. We employ spatial and temporal averaging to mitigate the noise and ensure adaptation is concentrated in consistently high error regions. First, temporal error indicator averaging is applied. The indicator is computed for each element on each snapshot, the indicators on each element are then averaged over time. Second, spatial averaging is applied over known statistically homogeneous directions. For example, in a turbulent channel, the streamwise and spanwise directions are statistically homogeneous. Therefore, slabs of elements at constant wall normal position are aggregated and the groups with the lowest error are adapted. We use the fact that the mesh is structured to ease the definition of these element groups. This technique ensures the adapted regions remain as stable as possible.

For each error indicator at each adaptation iteration, we will compute the error at each element and select approximately 20% of the elements with the worst error for adaptation. If there are known statistically homogeneous directions, elements in these directions are aggregated, their errors are summed, and compared with other sets of elements. For each test case, we have used structured grids with faces aligned in statistically homogeneous directions where applicable. This makes the definition of element groups simple and easy to implement. Since the super-resolution models operate on basis function coefficients, a different model is used for each input order. Since the input order of the network is fixed on both the element of interest and its neighbors, each neighboring element must be projected to the network's expected input order regardless of its original order.

We test both error indicators for two adaptive iterations on several test cases. The time integration order is held constant at 4, along with the time step for all cases. Each case

begins with $p = 3$ elements. We find that $p = 3$ is a good starting point for both low mesh resolution and reasonable starting accuracy. Adaptations average the error indicator over 20 iterations. Each adaptive iteration increments the order of the worse elements by two. After the adaptive iterations, we should be able to observe significant improvement in turbulent statistics using relatively few degrees of freedom.

We perform two adaptation iterations for each case. The number of adaptation iterations is limited by two main factors. The first is the ability of the network to accurately reconstruct flows at ever higher orders. We saw in Chapter 4 that as network size is increased, reconstruction quality plateaus and then declines. This limitation is due to the simple architecture of the network. As the input order is increased, the network's input size grows polynomially, and reconstruction quality will ultimately degrade. The second factor is that our code only supports odd polynomial orders, so the adapted order must increase by at least two at each iteration.

5.4 Unbiased Channel Test Cases

5.4.1 Geometry

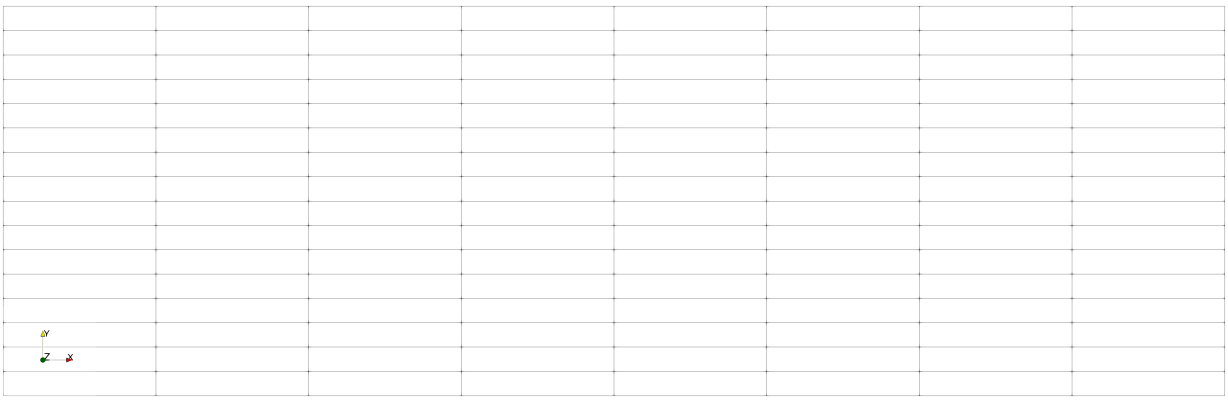


Figure 5.1: Uniform channel element outline. All elements are identical shape and aspect ratio.

In a turbulent channel mesh resolution will typically be focused near the walls where turbu-

lent kinetic energy is highest [88]. In addition, the chosen channel size should keep two-point velocity correlations at half domain distance in the periodic directions small [88, 73]. For our cases we have chosen to follow Moser *et al.* [89] with a domain measuring $2\pi\delta$ in the streamwise direction and $\pi\delta$ in the spanwise direction, where δ is the channel half height. Instead of biasing resolution toward the walls in our adapted cases, we have chosen to begin adaptation on a mesh with uniformly spaced elements in all directions. Starting with uniform elements allows the adaptation algorithm to determine a resolution distribution we can compare to prior knowledge of resolution requirements. Considering the $Re_\tau = 395$ case and comparing Table 5.1 to recommended wall-resolved LES resolutions in [55], the uniform channel meshes should be reasonably well resolved in the streamwise and spanwise directions but element spacing at and near the wall should be extremely lacking. These are approximate spacings found by dividing element size in wall units by the number of one dimensional degrees of freedom for that element.

Table 5.1: Approximate grid spacing in wall units for uniformly spaced channel mesh at $Re_\tau = 395$.

	Δx^+	Δz^+	Δy^+
N = 4	78	39	12
N = 8	39	20	6

The initial condition is a uniform velocity field with sine wave variation of various frequencies in all velocities and in all directions. This field is integrated forward in time until a linear profile of total shear stress, $\left(\overline{u'v'} - \mu\partial\bar{u}/\partial y\right)$, is achieved as discussed in [73]. Averaging over the streamwise and spanwise directions is employed to accelerate convergence of the statistical profiles. Once a low-order solution has reached statistically steady-state it is used to seed high-order solutions which undergo the same process with an improved initial condition. Each initial $N = 4$ solution is adapted for two iterations with approximately 20% of elements incremented by two polynomial approximation orders each iteration. While the error indicator is computed for each element, the adaptation takes advantage of the statistical streamwise homogeneity, spanwise homogeneity, and center-line mirror of the channel case.

This means all elements at a particular wall normal distance are aggregated when determining adaptation groups.

5.4.2 State Difference Error Indicator

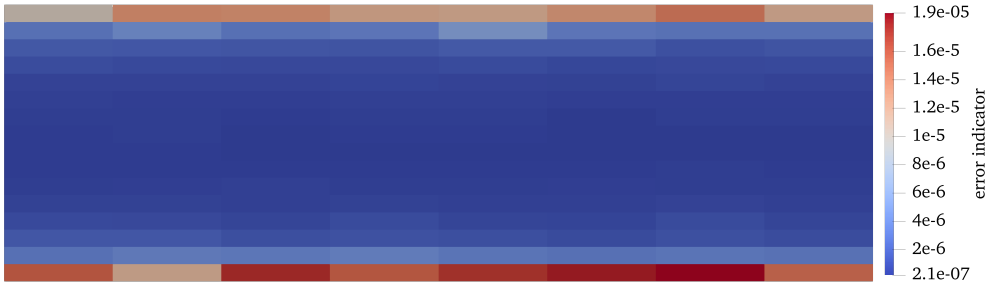
State Difference Indicator at $Re_\tau = 395$

case	degrees of freedom
uniform $p = 3$	65,536
uniform $p = 5$	221,184
uniform $p = 7$	524,288
adapted, state difference, iteration 1	104,448
adapted, state difference, iteration 2	161,792
adapted, weighted residual, iteration 1	104,448
adapted, weighted residual, iteration 2	161,792

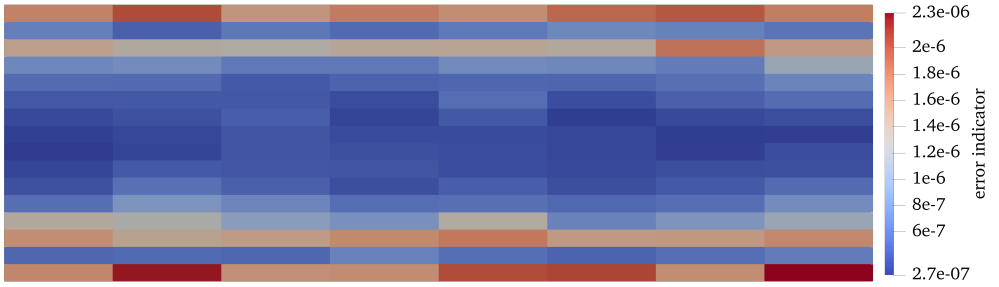
Table 5.2: Adapted degrees of freedom relative to uniform refinement for various cases of the uniform $Re_\tau = 395$ channel case.

Let us begin by discussing the $Re_\tau = 395$ results. The trends discussed here will largely generalize to the higher Reynolds number versions of this case. Error indicator values for each adaptation iteration along with the resulting order distribution for each iteration are shown in Figure 5.2. Selected turbulent statistics for uniform refinement, adaptation, and DNS data from literature, are reported in Figure 5.3.

Turning attention to the velocity profile in the top left of Figure 5.3, the uniform $p = 3$ solution has a bump at about $y^+ = 50$. This bump marks the end of the first element, such is the extreme under-resolution of this case near the wall. We expect that the adaptive technique will detect this under-resolution and correct it. Looking at the indicated error for the first adaptive iteration in Figure 5.2, we see that the most error is indicated near the wall. This high error then quickly and smoothly drops off toward the center of the channel. This pattern is consistent with the order distribution we expect from a properly set up channel case, this is a good sign. There appears to be a bias in indicated error toward the bottom of the channel. This has no influence on the adapted results since we are using the fact that

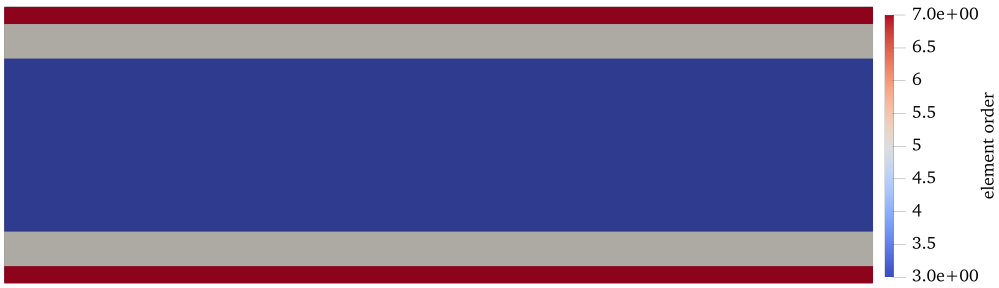


(a) Error indicator, first iteration



(b) Error indicator, second iteration

(c) Order distribution, first iteration



(d) Order distribution, second iteration

Figure 5.2: Error indicator and element order plots for an $Re_\tau = 395$ uniformly spaced channel using the state difference indicator.

this case is mirrored over the center line, so the elements on the top and bottom wall are adapted as a single group. The effect may simply be due to the fact that each plot in Figure 5.2 is a slice and not necessarily indicative of error at all spanwise locations. It may also be due to element orientation considerations.

Element orientation is a property of the mesh connectivity. Orientation information tells us how adjacent elements are rotated relative to the element of interest. During training and testing, we rely on a constant element orientation assumption. This has the downstream consequence that, for instance, if training on turbulent channel data, walls always occur on particular element faces. Velocity profile variation also only occurs in particular directions. There could be some slight bias in the indicated error since the network is not strictly direction agnostic. Nonetheless, we achieve reasonable results with the current setup.

With the error estimate decided for the first iteration, the adaptation routine targets the first two layers. Once again, this is expected behavior. For the second adaptation iteration, indicated error has significantly decreased overall. This is evidence corroborating some of the work in Chapters 3 and 4. In Chapter 3 we saw that a simple version of the state difference indicator does show decreasing indicated error as the initial solution resolution is increased. Without this property, the algorithm could unnecessarily adapt high-resolution flow-field regions. The convergence property holds in so far as reconstruction is accurate. In Chapter 4, we saw reconstruction quality deteriorate relative to Chapter 3. The drop in indicated error during adaptation could indicate reconstruction quality is still sufficient for adaptation.

For the second iteration, error remains high near the wall. Given the poor statistics shown in Figure 5.3, this appears to be a reasonable choice. Error remains low at the second layer off the wall, likely due to the adpatation in that region despite relatively low error to begin with. The third layer off the wall has relatively high error. This layer received no adaptation in the first iteration, so this pattern makes sense. After adapting again, the element order

next to the wall is moved to $p = 7$ with the next two layers at $p = 5$.

Now that we have seen reasonable adaptation patterns from the proposed process, we will take a look at the resulting statistics. The velocity profile in the upper left of Figure 5.3 shows the adapted cases moving in the correct direction toward the high-order results. Flow velocity in the center of the channel remains low relative to the $p = 7$ case. The source term driving the channel is constant for all cases. The unrefined region near the center of the channel should produce higher numerical dissipation than the near wall regions resulting in the velocity deficit. While the "bump" at the end of the first element in the $p = 3$ velocity profile is removed after the first adaptive iteration, the second iteration only slightly improves the velocity profile. This could be because after removing the extreme under-resolution near the wall, the primary error is the unrefined center region that cannot be removed without many adaptive iterations.

The extremely noisy first element is clearly visible in the streamwise root mean square profile in the upper right of Figure 5.3. The primary influence of adaptive and uniform refinement is simply controlling the wild variation in this element. Beyond the first element, the adaptive pattern is somewhat visible by kinks in the adapted profile. The first adapted profile has a kink at about $y^+ \approx 100$ and the second adapted profile has one at $y^+ \approx 150$. These kinks correspond to the edge of the adapted regions shown in Figure 5.2. The adapted profiles move relatively close to the $p = 7$ profile, but remain extremely noisy.

The wall normal root mean square profile is much better behaved than its streamwise counterpart. No significant noise is observed, just significant undershoot for the initial $p = 3$ solution. The first adaptation iteration significantly reduces the undershoot in the region below $y^+ \approx 100$, where the mesh has been adapted. While the near wall region shows improvement, the remainder of the profile shows more undershoot than the baseline solution. The second adaptation iteration shows the same effect. The near wall region is improved while far from the wall the profile degrades. This degradation could be another example of

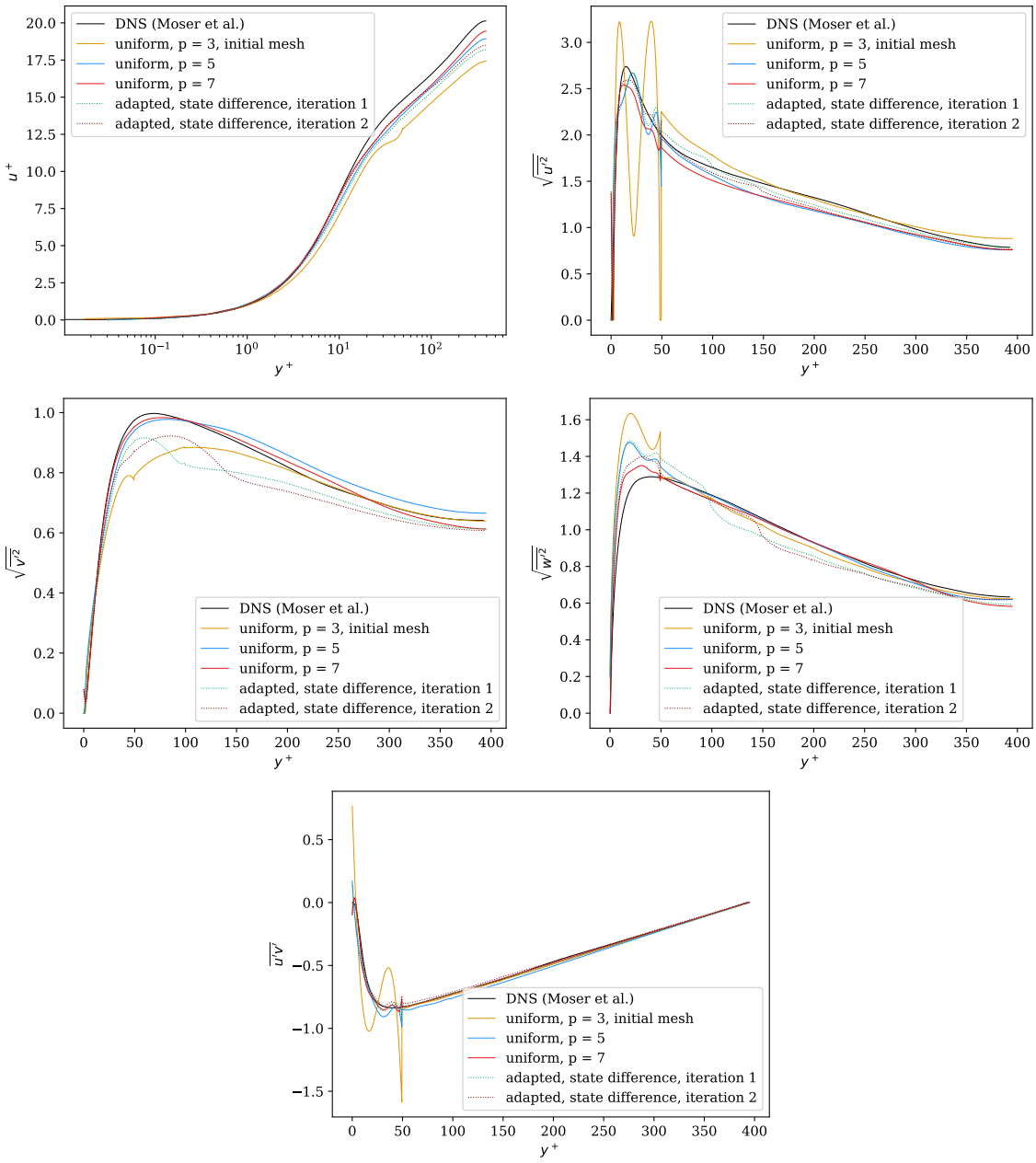


Figure 5.3: Turbulent statistics comparing two iterations of the state difference error indicator with uniform refinement at $Re_\tau = 395$.

jarring resolution changes within the computational domain hindering the accuracy of the solution.

The spanwise root mean square profile continues themes in the previous profiles. The adapted region is clearly visible, with the adapted profiles showing an almost "stepped" character moving from left to right. After the adapted region, we once again see degradation in the profile relative to the initial solution. The adaptation clearly controls the extreme error in the first element as expected. It does not reach the same accuracy as uniform refinement near the wall despite having the same order of accuracy in that region, showing the negative influence of unadapted central channel regions.

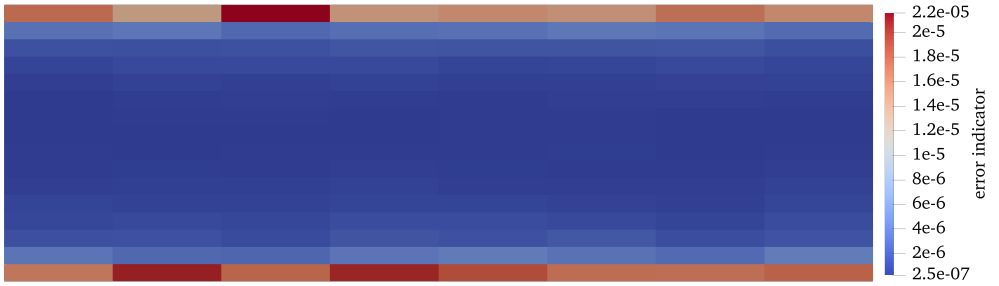
Finally, the Reynolds shear stress is the easiest statistic to capture. Even the initial profile is nearly perfectly accurate, but for the first element. The increased order at the first element for the adapted cases cleans up the first element error nicely. The adapted profiles are similar to the uniformly refined $p = 7$ profile.

State Difference Indicator at $Re_\tau = 590$

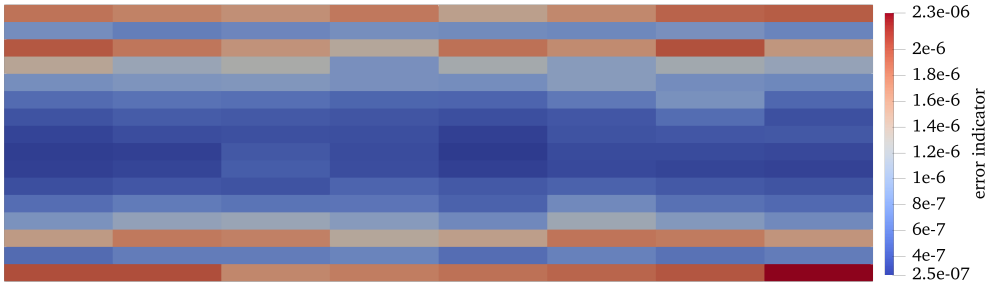
case	degrees of freedom
uniform $p = 3$	65,536
uniform $p = 5$	221,184
uniform $p = 7$	524,288
adapted, state difference, iteration 1	104,448
adapted, state difference, iteration 2	161,792
adapted, weighted residual, iteration 1	104,448
adapted, weighted residual, iteration 2	161,792

Table 5.3: Adapted degrees of freedom relative to uniform refinement for various cases of the uniform $Re_\tau = 590$ channel case.

In this section we test adaptation on the uniform channel at $Re_\tau = 590$. Error indicators and element order distributions for the two adaptation iterations are shown in Figure 5.4. It appears running the same case at this higher Reynolds number has not changed the results at all. The error indicator for the first iteration still shows high error near the wall with

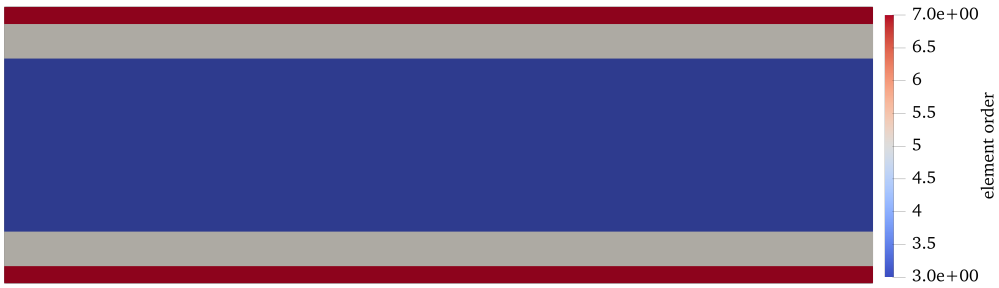


(a) Error indicator, first iteration



(b) Error indicator, second iteration

(c) Order distribution, first iteration



(d) Order distribution, second iteration

Figure 5.4: Error indicator and element order plots for an $Re_\tau = 590$ uniformly spaced channel using the state difference indicator.

smooth drop off toward the center of the channel. For the second iteration, the error indicator shows the same pattern with high error near the wall, a drop where the elements have been adapted, and a rise once again where the elements have not been altered. The element order distribution also remains identical to the $Re_\tau = 395$ case.

The similarity in these results is probably a good sign. Nothing fundamentally changes about the character of the channel solution between $Re_\tau = 395$ and $Re_\tau = 590$. The most difficult to resolve region is still just off the wall, and the center of the channel is easy to capture with relatively large eddies.

The statistics in Figure 5.5 also tell a similar story. The bump anomaly in the first element is eliminated after adaptation. Interestingly, it takes both adaptation iterations to recover performance near the center of the channel after eliminating the velocity profile bump. Two adaptation iterations easily eliminate the error in the streamwise root mean square profile. The solution becomes roughly consistent with $p = 7$ uniform refinement on this metric using about 70% fewer degrees of freedom, referencing Table 5.3. The other two root mean square profiles show less accuracy than their streamwise counterpart. The edges of the adapted region are also clearly visible on these plots. Reynolds shear stress is also easily captured, seeing as almost all error resides in the first element of the wall and that element is heavily adapted.

State Difference Indicator at $Re_\tau = 950$

case	degrees of freedom
uniform $p = 3$	65,536
uniform $p = 5$	221,184
uniform $p = 7$	524,288
adapted, state difference, iteration 1	104,448
adapted, state difference, iteration 2	161,792
adapted, weighted residual, iteration 1	104,448
adapted, weighted residual, iteration 2	161,792

Table 5.4: Adapted degrees of freedom relative to uniform refinement for various cases of the uniform $Re_\tau = 950$ channel case.

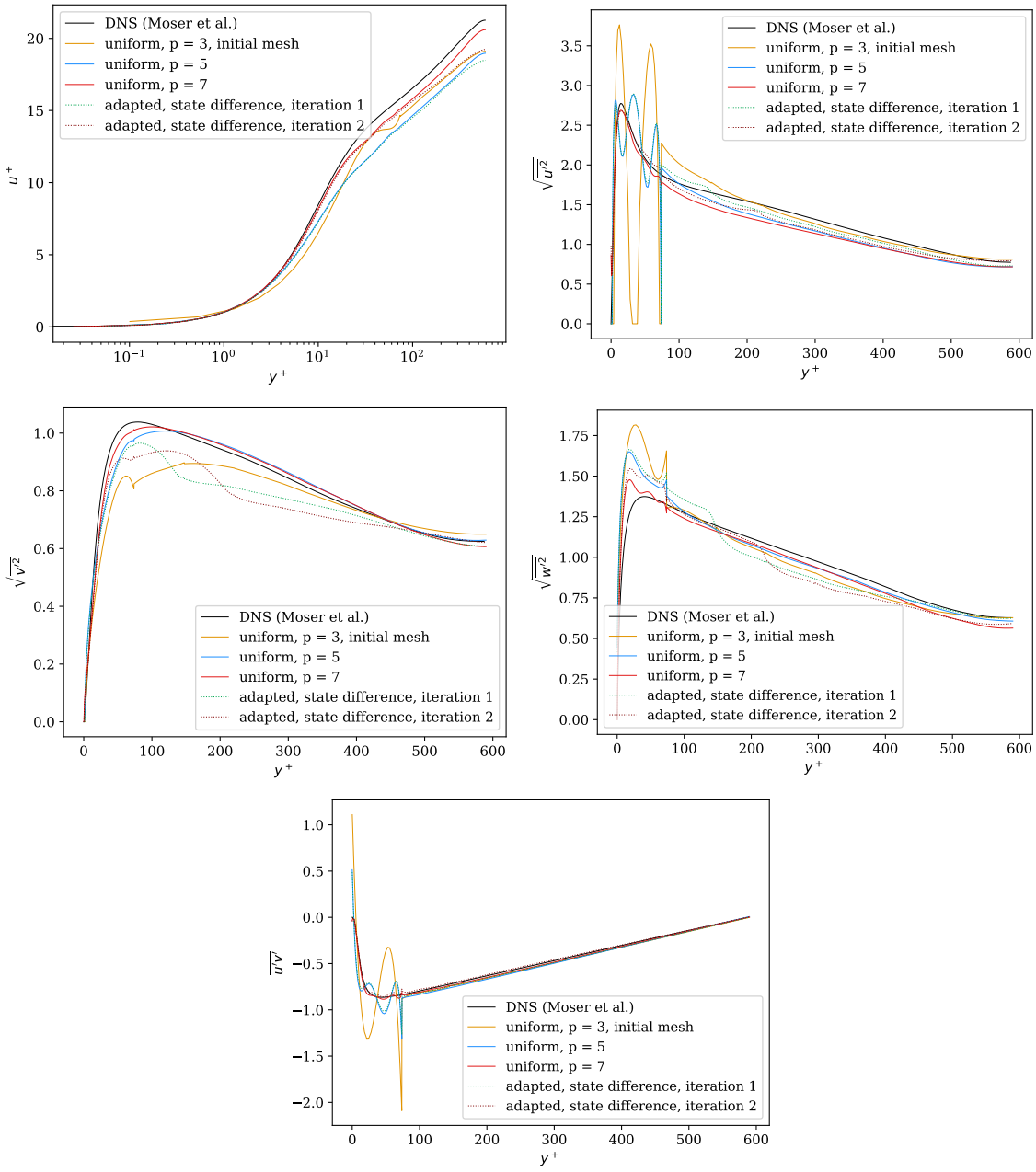


Figure 5.5: Turbulent statistics comparing two iterations of the state difference error indicator with uniform refinement at $Re_\tau = 590$.

Once again we run the same adaptive iterations on the same turbulent channel with no preexisting element refinement pattern. The adapted error indicators and order distributions are shown in Figure 5.6. The relatively high error on the lower wall for the first adaptation iteration is persistent, even in this case. Since the same network has been used for each adaptation, this probably confirms that this network has a bias. The adapted pattern once again remains exactly the same as the $Re_\tau = 395$ and $Re_\tau = 590$ cases.

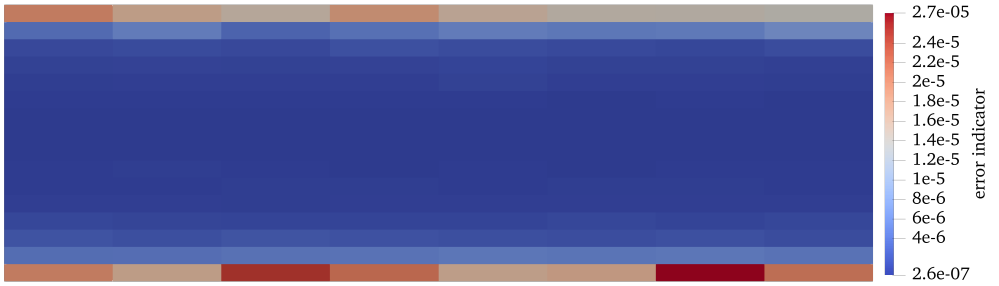
Despite a lack of DNS data for the $Re_\tau = 950$ case, we can still compare the adapted results against uniform refinement. Overall, the profiles after two adaptation iterations compare favorably with uniform refinement at $p = 5$. Once again that is at fewer degrees of freedom according to Table 5.4. The velocity and streamwise root mean square profiles show promising results, each comparing quite close to the $p = 7$ solution. The root mean square profiles in the other directions are less favorable, once again repeating trends seen previously. The noise in all profiles has significantly increased, this is expected on the same mesh at higher Reynolds number. This accounts for the persistent noise in the Reynolds shear stress profile, $p = 7$ on the first element is not sufficient to capture the profile under either uniform or adapted refinement.

5.4.3 Entropy-Adjoint-Weighted-Residual Error Indicator

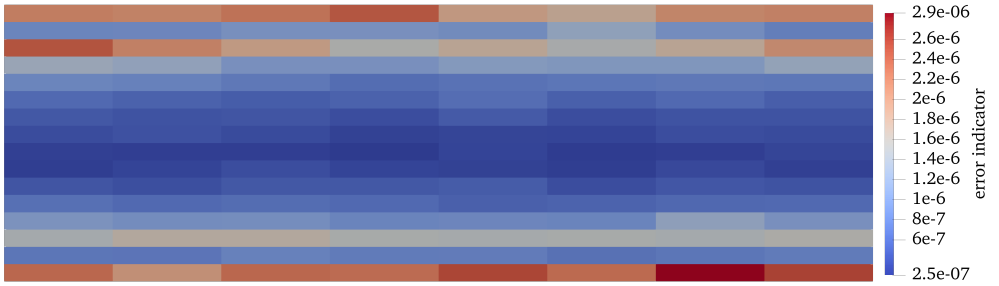
Entropy-Adjoint-Weighted-Residual Indicator at $Re_\tau = 395$

The adjoint weighted residual indicator behaves completely differently than the state difference indicator. For the $Re_\tau = 395$ case, error indicator and adapted order plots are shown in Figure 5.8. Focusing on the error indicator plot for the first adaptive iteration, it is much noisier than the state difference version. The regions of high indicated error are generally concentrated near the walls, but the signal is not nearly as strong as the other indicator. The spatial averaging will help with this noise to some degree when adapting the mesh.

We see that the order distribution after adapting one time brings two layers to $p = 5$, but

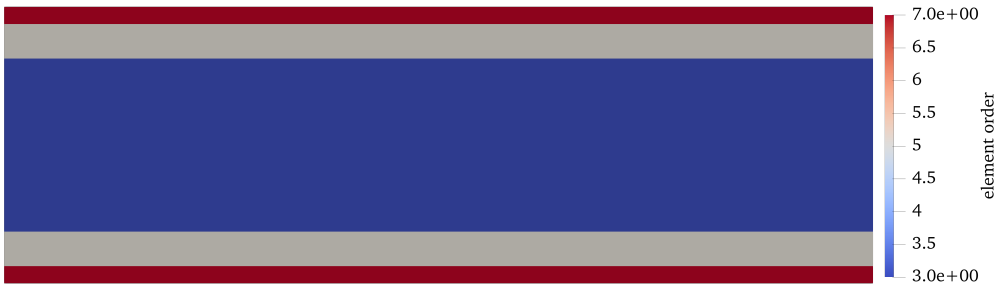


(a) Error indicator, first iteration



(b) Error indicator, second iteration

(c) Order distribution, first iteration



(d) Order distribution, second iteration

Figure 5.6: Error indicator and element order plots for an $Re_\tau = 950$ uniformly spaced channel using the state difference indicator.

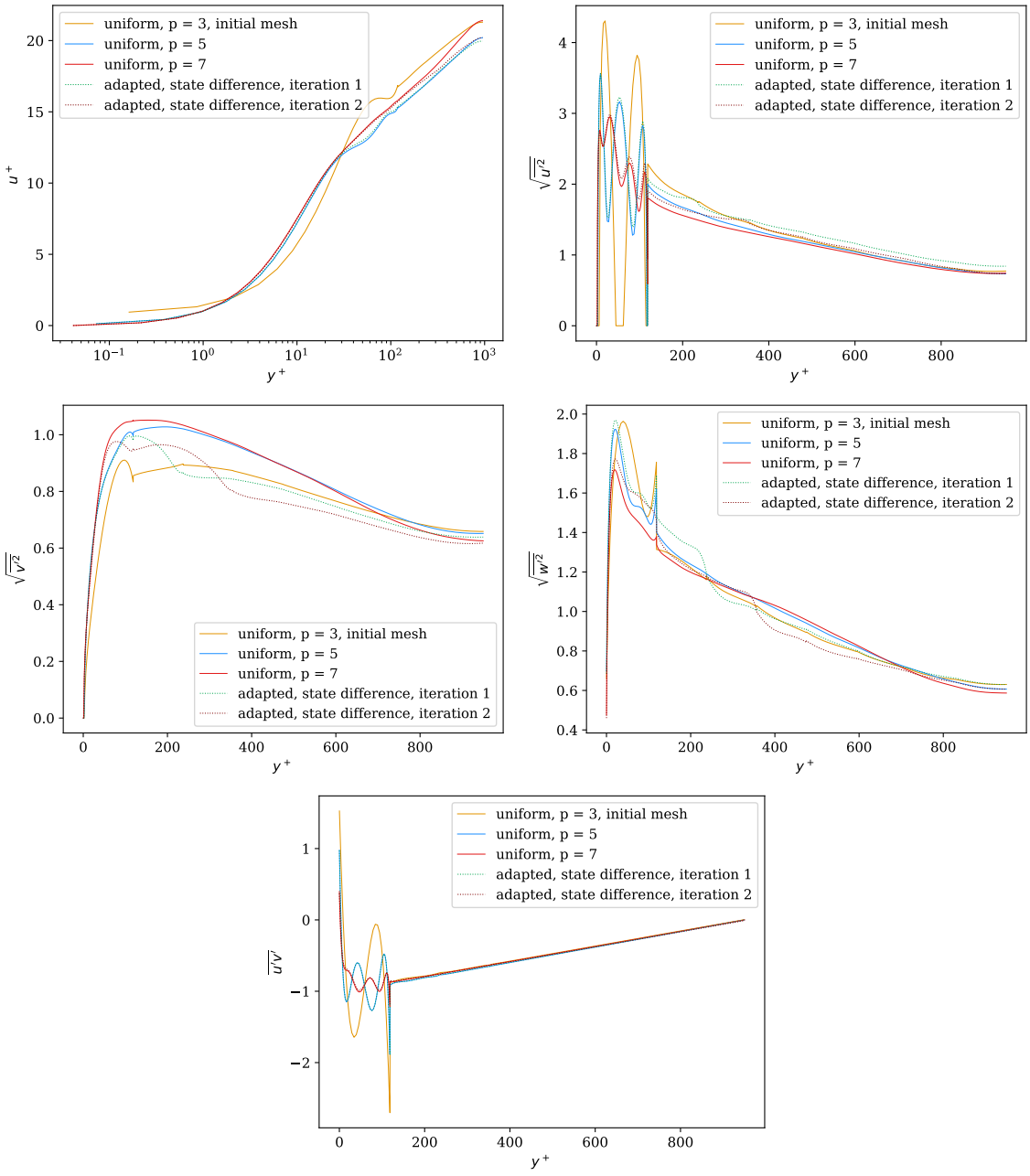
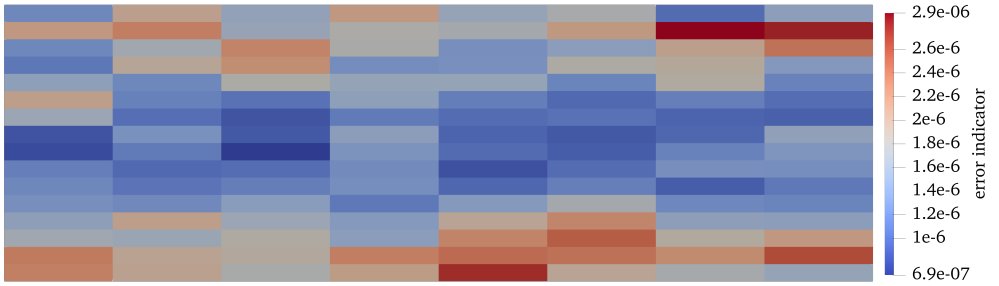
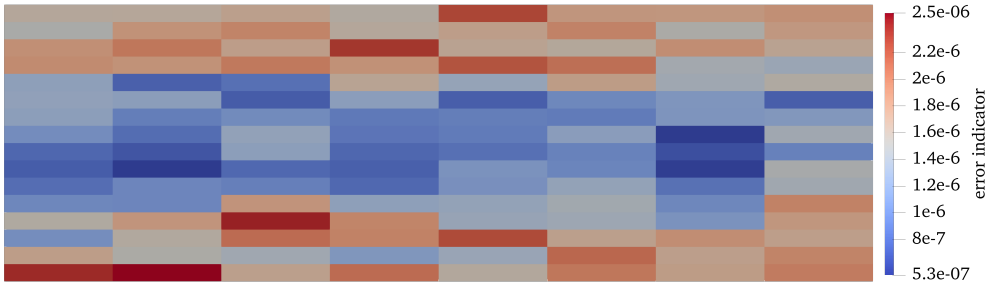


Figure 5.7: Turbulent statistics comparing two iterations of the state difference error indicator with uniform refinement at $Re_\tau = 950$.

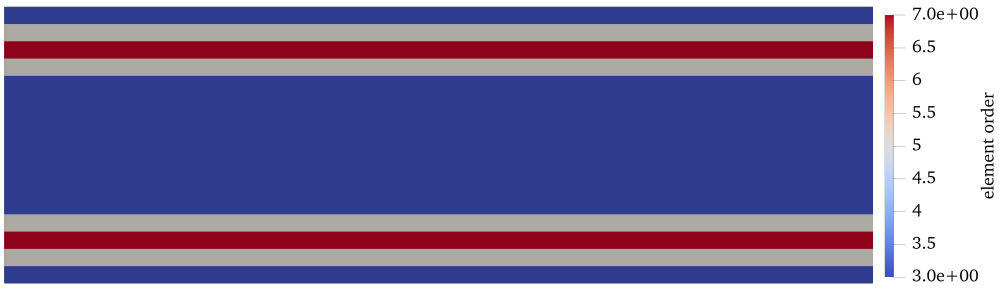


(a) Error indicator, first iteration



(b) Error indicator, second iteration

(c) Order distribution, first iteration



(d) Order distribution, second iteration

Figure 5.8: Error indicator and element order plots for an $Re_\tau = 395$ uniformly spaced channel using the adjoint weighted residual error indicator.

misses the first element on the wall. Previously, we established that the error on the elements adjacent to the wall is extremely high. This test case is even designed to ensure that the error near the wall is extremely high. This oversight should be a serious miss on the part of this indicator, we will confirm this when discussing the statistics.

The error indicator for the second adaptive iteration shows a similar noisy pattern to the first iteration. The focus continues to be near the walls, but the noise is extremely high. The mid point of the noise pattern is also clearly somewhat off the wall. This results in the first layer of the wall remaining unrefined for a second iteration in a row. Once again, this should bode poorly for the statistical results.

Focusing on the velocity profile in the upper left of Figure 5.9, our suspicions based on the adapted orders are confirmed. We observe a degradation in the velocity profile for both adapted cases. It is clear that targeting the first element is critical. This makes sense considering that in the channel, the only geometry is the wall. Resolution errors near the wall will propagate throughout the flow domain, hence the necessity to refine near channel walls. The slight velocity profile degradation, as opposed to simple stagnation, may be due to the effect of jarring resolution change in the computational domain.

The streamwise root mean square profile shows the same story. The unrefined first layer allows little improvement in the overall profile. The extreme noise of the initial $p = 3$ solution only slightly changes with refinement focused farther into the channel.

The wall normal root mean square profile is a little more interesting. Once again, we can roughly see the adaptation pattern in the locations where undershoot is most reduced. The first adapted profile has a bump between $y^+ \approx 50$ and $y^+ \approx 150$ while the second adapted profile peaks between $y^+ \approx 50$ and $y^+ \approx 200$. We continue to see degradation in the profile for the remainder of the channel. The real failing here appears to be the inability to capture the peak around $y^+ \approx 60$ correctly without adapting the first element.

The spanwise root mean square profile continues to be the statistic where the adapted pattern

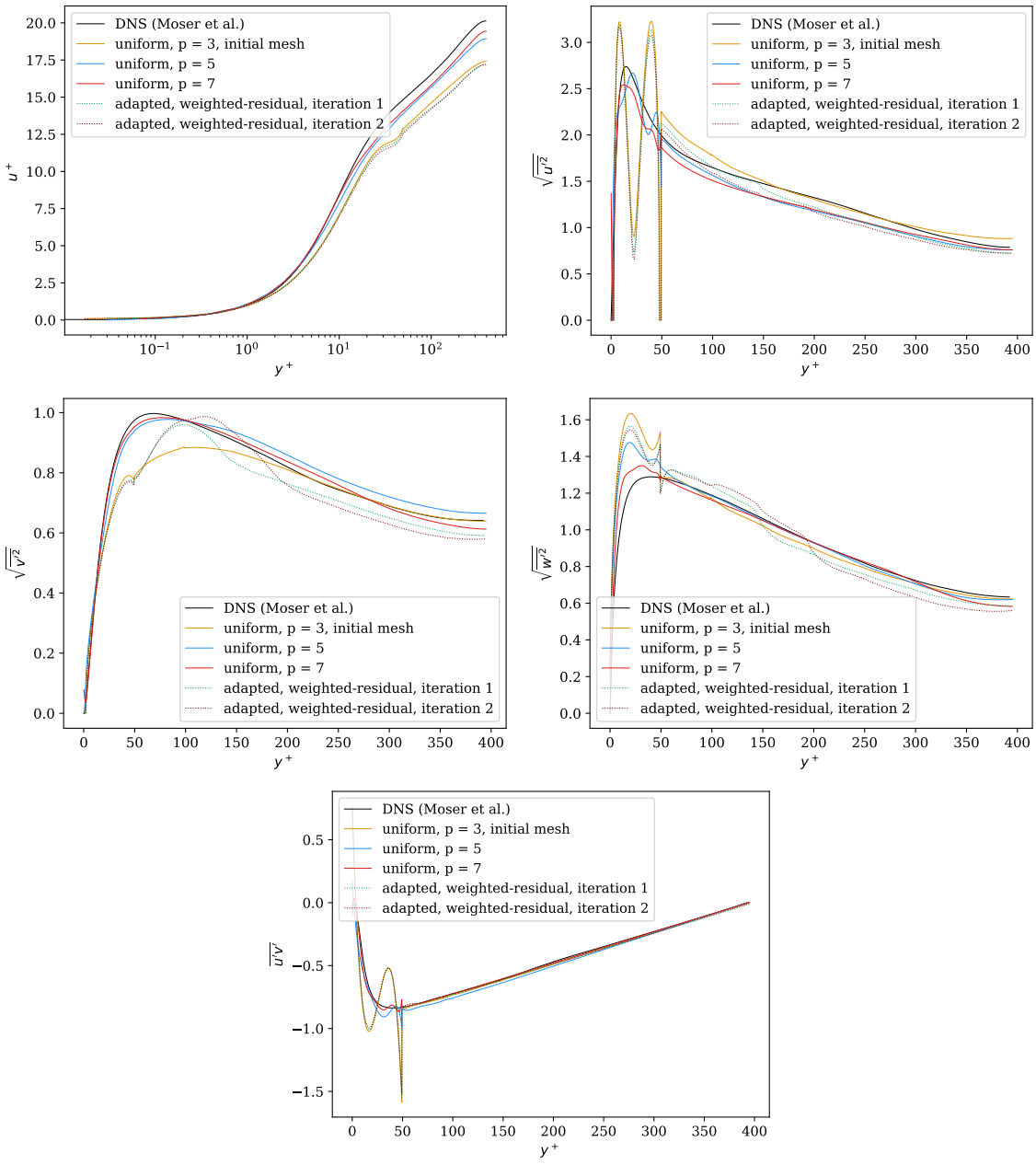


Figure 5.9: Turbulent statistics comparing two iterations of the entropy-adjoint-weighted-residual error indicator with uniform refinement at $Re_\tau = 395$.

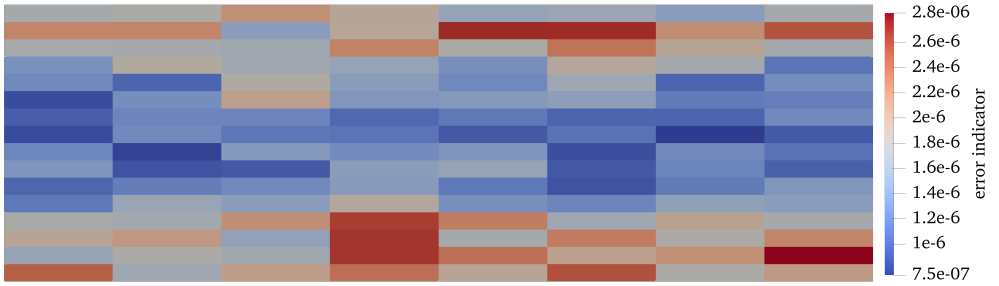
is easiest to see. Even the $p = 7$ peak in the order distribution after two adaptations is visible in the profile. The region near the wall improves slightly with help from adaptation elsewhere, but error remains very high.

The Reynolds shear stress plot fundamentally does not change. Almost all the error, even in the initial $p = 3$ solution, is in the first element. Since that element is not touched, the adapted profiles are effectively unchanged from the initial profile.

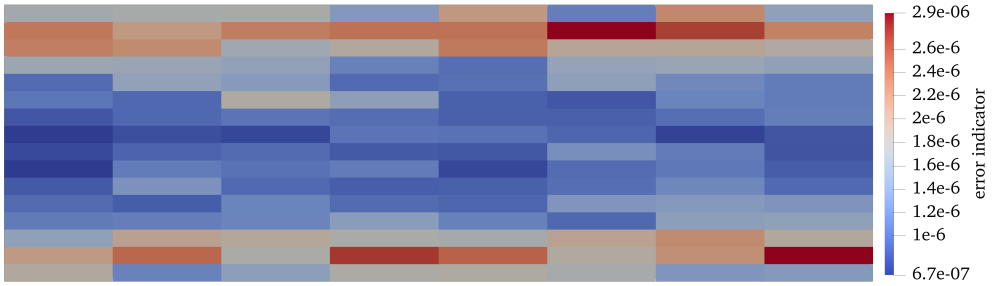
Entropy-Adjoint-Weighted-Residual Indicator at $Re_\tau = 590$

While the adapted results were poor at $Re_\tau = 395$, they could still improve at higher Reynolds numbers. The error indicator and adaptation patterns are shown in Figure 5.10. While the error distribution in the first element appears similar to the $Re_\tau = 395$ case, this time the first element off the wall is adapted. This may simply be due to slight differences in the noise pattern tipping the balance in favor of the first element off the wall instead of the third. It is also possible that the region of high indicated error has truly shifted closer to the wall. The error indicator pattern for the second iteration may lend some credibility to this idea. This time, the error distribution has clearly tightened near the wall relative to the $Re_\tau = 395$ distribution. Highest error is indicated just off the wall, leading to the same pattern seen for $Re_\tau = 395$ shifted one element toward the wall.

While the statistics in Figure 5.11 do show some improvement this time, they are far from the results of the state difference indicator. The velocity profile performs well close to the wall, but falls away shortly thereafter. Root mean square profiles are also improved. The wall normal direction shows the most promise, where the peak is nearly matched by the adapted result. The spanwise results continue to show the stepped adaptation pattern. While the character is the same as in the $Re_\tau = 395$ results, the steps in the profile have shifted toward the wall, reflecting the change in the adaptation pattern. The Reynolds shear stress shows increased accuracy as expected from refinement near the wall.



(a) Error indicator, first iteration



(b) Error indicator, second iteration

(c) Order distribution, first iteration

(d) Order distribution, second iteration

Figure 5.10: Error indicator and element order plots for an $Re_\tau = 590$ uniformly spaced channel using the adjoint weighted residual error indicator.

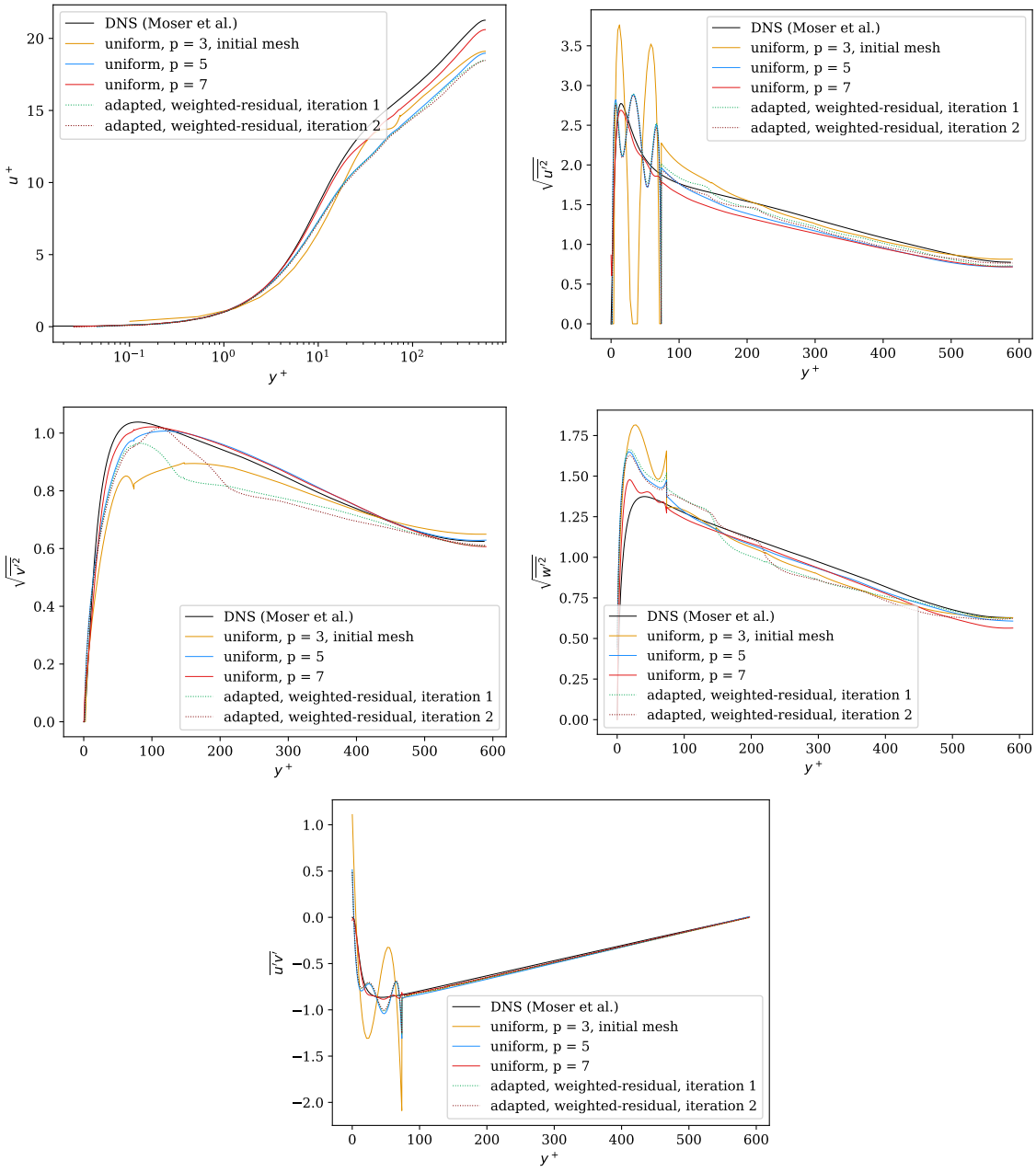


Figure 5.11: Turbulent statistics comparing two iterations of the entropy-adjoint-weighted-residual error indicator with uniform refinement at $Re_\tau = 590$.

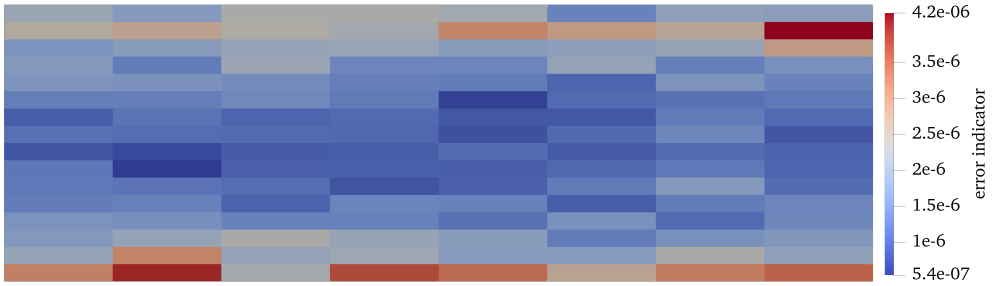
The $Re_\tau = 590$ results appeared to show a trend where higher error is indicated near the walls relative to the initial $Re_\tau = 395$ test cases. It appears that the $Re_\tau = 950$ test cases continue the same trend. The error indicator and adapted element patterns are shown in Figure 5.12. While still very noisy, the indicated error in the first iteration is beginning to look more like the state difference error indicator. As in the $Re_\tau = 590$ case, the first two elements off the wall are adapted. The error indicator for the second iteration appears to continue the trend where error is indicated increasingly close to the wall. It is not yet close enough to adapt the first element off the wall a second time, instead the second iteration has the same adapted order pattern as the $Re_\tau = 590$ case.

Entropy-Adjoint-Weighted-Residual Indicator at $Re_\tau = 950$

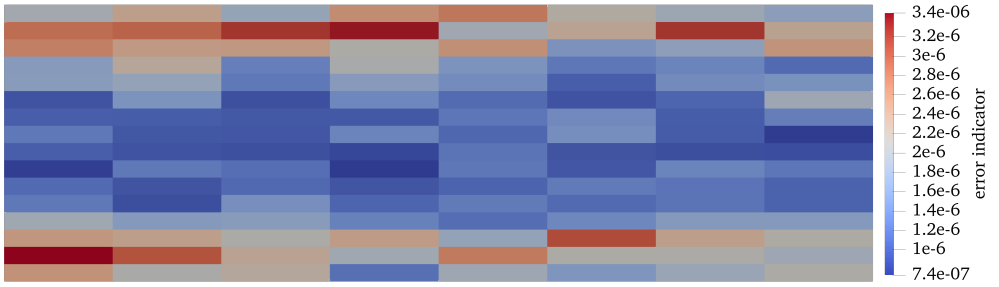
Statistical profiles for the $Re_\tau = 950$ test cases are shown in Figure 5.12. Results are broadly similar to the $Re_\tau = 590$ versions, but with significantly more noise due to under-resolution. This is to be expected since the adaptation pattern is the same in both cases.

The relatively poor performance of the entropy-adjoint-weighted-residual is unexpected since, at least on the surface, it is the more sophisticated of the two. Several observations from present and prior testing may shed light on its performance. In the definition of the weighted residual error indicator, the absolute value is taken after all residual weighting has taken place for the entire element. This means that cancellation internal to an element is allowed. Past versions of the state difference error indicator that allowed internal cancellation also exhibited noisy patterns of indicated error. If we assume that the residual is relatively constant over the element, cancellation from the adjoint difference may be contributing to the excess noise. This idea may be backed up by the fact that the super-resolution network only predicts corrections to the baseline state. This correction likely has a mean near zero, which could explain the noisy indicated error pattern.

The averaging behavior may also be to blame, or at least exasperate the possible cancellation issue. Previous works, such as that of Bassi *et al.* in [10], have averaged the entropy-adjoint

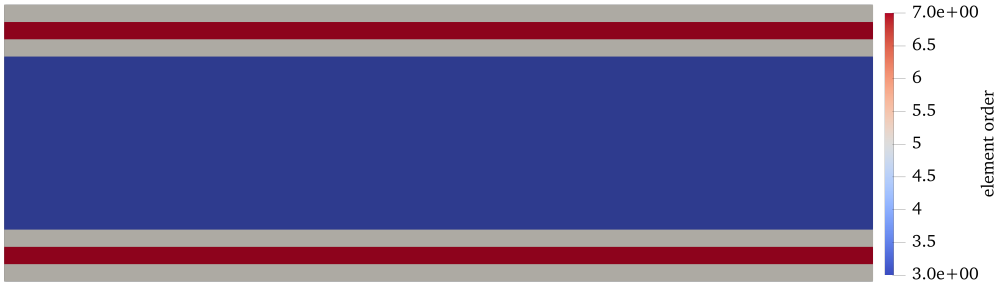


(a) Error indicator, first iteration



(b) Error indicator, second iteration

(c) Order distribution, first iteration



(d) Order distribution, second iteration

Figure 5.12: Error indicator and element order plots for an $Re_\tau = 950$ uniformly spaced channel using the adjoint weighted residual error indicator.

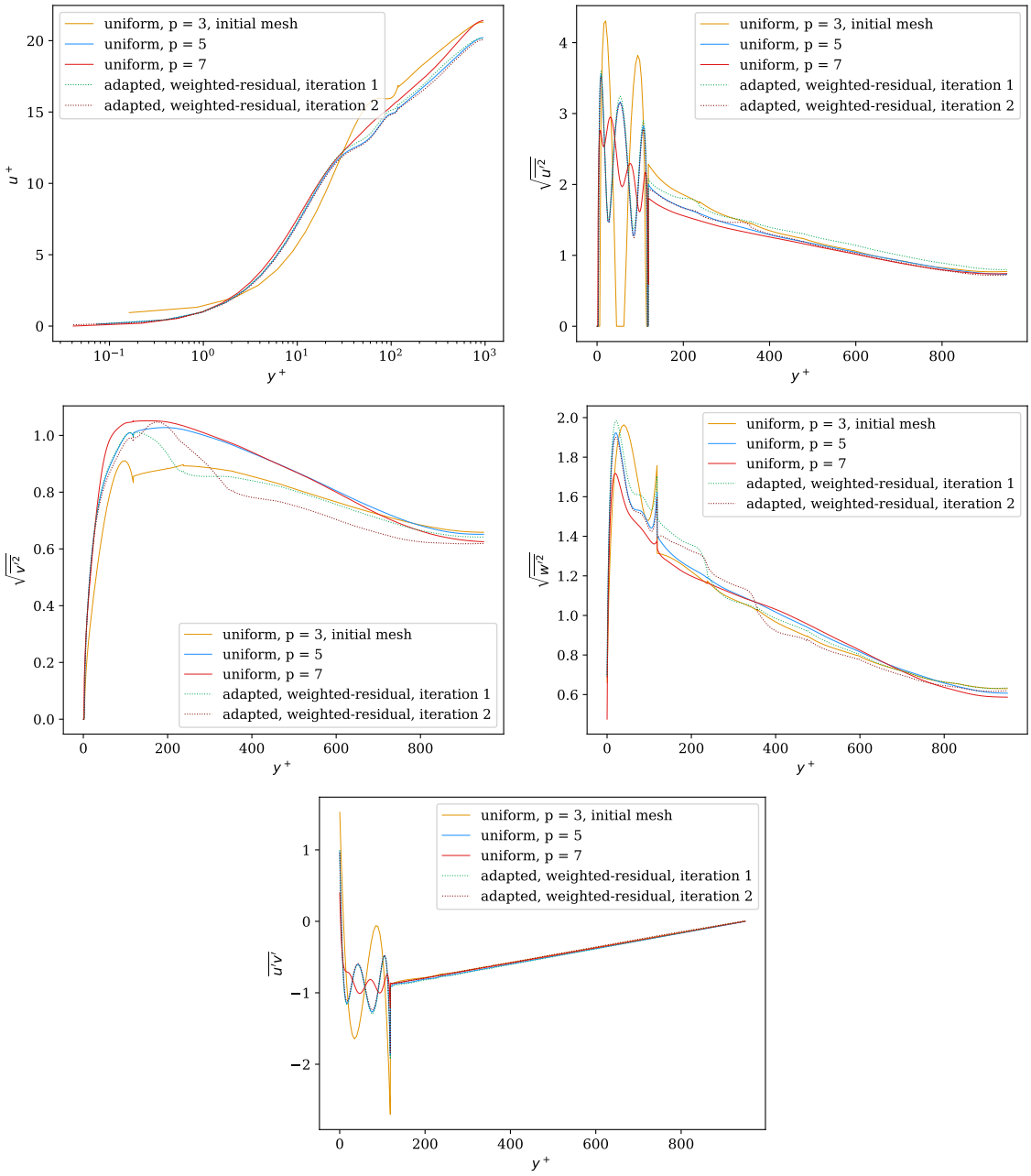


Figure 5.13: Turbulent statistics comparing two iterations of the entropy-adjoint-weighted-residual error indicator with uniform refinement at $Re_\tau = 950$.

and residual over time before attempting to compute the error indicator. That is, as opposed to computing the error on each snapshot and only then averaging over time. Switching to this averaging strategy could yield much improved results.

We observe that the region of highest indicated error moves increasingly close to the wall as Reynolds number increases for the weighted residual indicator. The most plausible explanation for this trend could be the influence of the residual on the error indicator. As the Reynolds number increases, the solution becomes effectively increasingly under-resolved as long as the mesh stays the same. This would lead to a higher residual when the projected coarse space is evaluated, possibly explaining the trend.

5.5 Comparison With Velocity Gradient Error Indicator

(a) Order distribution, first iteration

(b) Order distribution, second iteration

Figure 5.14: Element order plots for an $Re_\tau = 395$ uniformly spaced channel using a velocity gradient based indicator.

With the success of the state difference indicator on the channel test case, it can now be tested

against a more reasonable procedure than uniform refinement. For this, we have chosen a simple gradient-based indicator that uses the average flow velocity. Using the average flow velocity, we can reason about the adaptation pattern without implementing it in the form of an element-wise error indicator. Since the turbulent channel flow is homogeneous in the streamwise and spanwise directions, the average velocity gradient in those directions is zero. This leaves the wall-normal direction with non-zero average velocity gradient. We know that the channel walls are no slip, so the velocity there must be zero. We also know that velocity near the channel center is relatively constant, so there must be a rapid rise in velocity near the walls. It is reasonable, then, that the regions of highest velocity gradient will be near the walls, with regions of low velocity gradient near the center of the channel. If we assume a 20% fixed fraction adaptation strategy where all statistically homogeneous directions are adapted together, we can conclude that the first two layers will be adapted at each iteration. This adaptation pattern is shown in Figure 5.14.

case	degrees of freedom
uniform $p = 3$	65,536
adapted, velocity gradient, iteration 1	104,448
adapted, velocity gradient, iteration 2	180,224
adapted, state difference, iteration 1	104,448
adapted, state difference, iteration 2	161,792

Table 5.5: Adapted degrees of freedom relative to velocity gradient-based refinement for various cases of the uniform $Re_\tau = 395$ channel case.

Figure 5.15 shows comparison of turbulent statistics between two adaptations of the gradient indicator, and the state difference indicator. Each adaptation begins from the solution with $p = 3$ everywhere. The first adaptation iteration for both indicators is identical, therefore the results in the statistics only differ in their averaging. Each indicator handles the second adaptation iteration differently. The difference indicator spreads the adapted region by refining the third layer of elements off the wall to $p = 5$, while only taking the element on the wall to $p = 7$. In contrast, the gradient based indicator always refines the first two layers off the wall, leading to two layers at $p = 7$. This means the gradient based indicator uses

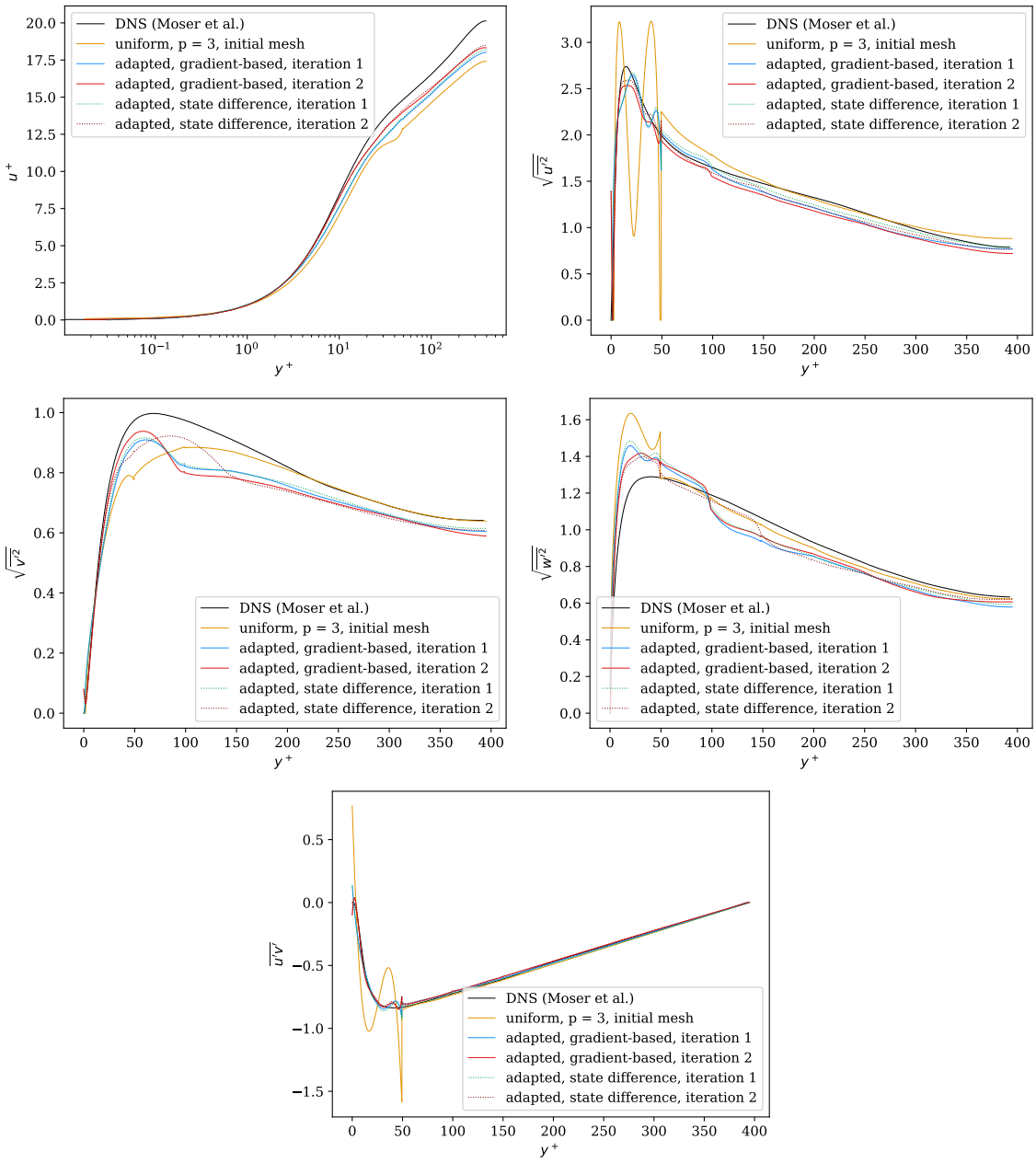


Figure 5.15: Turbulent statistics comparing two iterations of the state difference error indicator with two iterations of a velocity gradient indicator at $Re_\tau = 395$.

more degrees of freedom as seen in Table 5.5.

Identical velocity profile results in the first adaptation iteration are followed by nearly identical results in the second. It appears the difference in near-wall refinement pattern makes little difference on this case, the fact that the wall is refined is enough. The streamwise root-mean-square profile also shows little difference in the second adaptation iteration. Both methods control the severe near-wall under-resolution. Differences appear in the wall-normal and spanwise root-mean-square profiles. These have been the profiles where the details of the adapted patterns are easiest to see, and this case is no exception. The gradient-based indicator shows better results near the wall as expected from the adapted pattern, but worse results closer to the channel center where it has not refined. The root-mean-square plot once again has all error in the first element, this error is controlled by high-order near the walls for both indicators.

The performance of the state difference and gradient-based indicators has been nearly identical. The refinement pattern for both indicators is similar, focusing on the near-wall region. Adapted patterns differ on the second iteration, where the gradient-based indicator continues to concentrate resolution closer to the wall. The fixed-fraction adaptation strategy means that the gradient-based indicator's consistent concentration of resolution near the wall leads to increased cost. Based on these results, we may think of the super-resolution-based error indicators as a type of feature indicator. The regions of high velocity gradient and high reconstruction correction overlap significantly. It is more efficient however to spread resolution over a larger area as is the case in the super-resolution-based adaptation.

5.6 Periodic Hill Test Case

5.6.1 Geometry and Case Setup

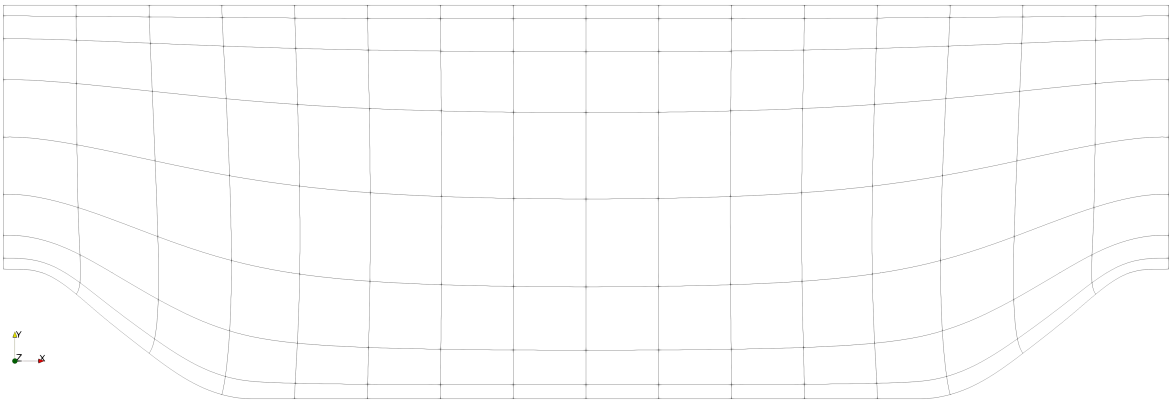


Figure 5.16: Periodic hill geometry. Elements are cubic to comply with spline geometry definition.

Almeida *et al.* [3] experimentally investigated flow over periodic hills in the early 1990s. The geometry was subsequently modified for a streamwise periodic computational setting by Mellen *et al.* [87]. Subsequent LES and DNS numerical investigations of the geometry at various Reynolds numbers have been conducted by Temmerman and Leschziner [110], Breuer *et al.* [18], Balakumar [7], and many others. The geometry used in the present study is shown in Figure 5.16. It consists of a single hill split at the streamwise periodic boundary of the computational domain. The hill is curved, defined as a spline between several lower surface points. The geometry is spanwise periodic, and both the upper and lower surfaces are no slip walls. This geometry is a classical case of flow separation and recirculation, which occurs after the first hill.

The mesh used for the present test cases is the low resolution mesh used by Diosady and Murman [32] for their DNS of the case for $Re_b = 10,595$. The resolution is $16 \times 8 \times 8$ elements in the streamwise, spanwise, and wall normal directions respectively. At this resolution the geometry definition is inexact. An exact definition would use third order curved elements and place element boundaries exactly at spline interpolation points. However, the inaccuracy

incurred should be extremely small, much smaller than the error due to under-resolution before adaptation. We have chosen this relatively low resolution to give poor results at the $p = 3$ initial order, and reasonably accurate results at $p = 7$. This will give adaptation the opportunity to show a significant improvement in turbulent statistics.

While the resolution is fixed by the mesh, we still need to set the Reynolds number to define the case. Periodic hill cases are driven by a body force. We maintain a constant body force, and this body force must be chosen correctly to achieve the correct flow condition. Periodic hill cases are defined by their bulk Reynolds number Re_b, where the length scale is the hill height and the velocity scale is the bulk velocity u_b, the average velocity at $x/h = 0$. In our case, we have fixed the kinematic viscosity and vary the body force to achieve the correct u_b and Re_b. This forcing study was done on a mesh with double the test mesh's element count in each dimension ($32 \times 16 \times 16$ elements) and polynomial order 7. While not a DNS resolution, this higher resolution ensures dissipation does not skew the forcing parameter to overcome the higher dissipation of a lower resolution case.

For all uniformly refined and adapted cases, the strength of the forcing parameter remains constant at the value discovered from the high-resolution simulation. In doing so, we will be able to see the effects of dissipation clearly. Low resolution simulations will have lower velocities than their higher resolution counterparts and refinement should monotonically approach DNS results. This will make visual distinction between more and less dissipative cases clear in the statistical profiles. When presenting statistical profiles they are also typically normalized using u_b. For all profiles presented below, the same normalizing $u_b = 0.2$ value will be used, once again to preserve intuitive convergence behavior in the final plots.

5.6.2 Adapted Results

Following are results for the state difference and entropy-adjoint-weighted-residual error indicators. Each indicator is run for two adaptive iterations on the $Re_b = 2800$ turbulent

channel with $16 \times 8 \times 8$ elements. Each adaptation begins from a uniform $p = 3$ solution and ends with mixed $p = 3$, $p = 5$, and $p = 7$ elements. Adaptations are compared with a series of uniform refined cases at $p = 3$, $p = 5$, and $p = 7$. Turbulent statistics at various stations are presented compared with DNS data from Breuer *et al.* [18]. For each adaptive order change, a burn time of approximately 50 flow through times is used to approach statistical steady-state before averaging begins. Turbulent statistics are then averaged over a further 50 flow through times for all cases. Statistics are then averaged in the spanwise direction to produce the final statistical profiles at each station.

Statistical profiles at various stations are shown in Figures 5.17, 5.18, 5.19, and 5.20 for x/h equal to 0.05, 2, 4, and 7 respectively. It should be noted from the beginning that there is some error incurred in the calculation of higher order statistics, especially at low resolutions. This may be noticed in small deviations from zero at the walls for the higher order statistics. This is because the calculation used assumes the fluctuating velocity component averages out to exactly zero, which is not strictly the case with finite averaging time. There could also be some influence of weakly enforced boundary conditions in busy flow-field areas. Nonetheless, there is strong agreement with the DNS results of Breuer *et al.* [18] provided on all plots, and only small deviations are observed at the walls.

A glace at each velocity profile shows the expected convergence behavior with increasing resolution. The lower resolution cases are more dissipative, leading to lower averaged velocities for the same body force. The uniformly refined $p = 7$ solution does not match the DNS data exactly, but this should be expected since it it still severely under-resolved. There is a large jump in accuracy between the $p = 3$ and $p = 5$ solutions, and a relatively small jump from $p = 5$ to $p = 7$. This is also expected, since we are effectively observing the convergence in linear scale, when error should decrease exponentially with increasing order. Focusing only on the velocity profiles, it appears the weighted residual indicator is performing better after two adaptive iterasions. It consistently approaches close to the $p = 5$ solution using

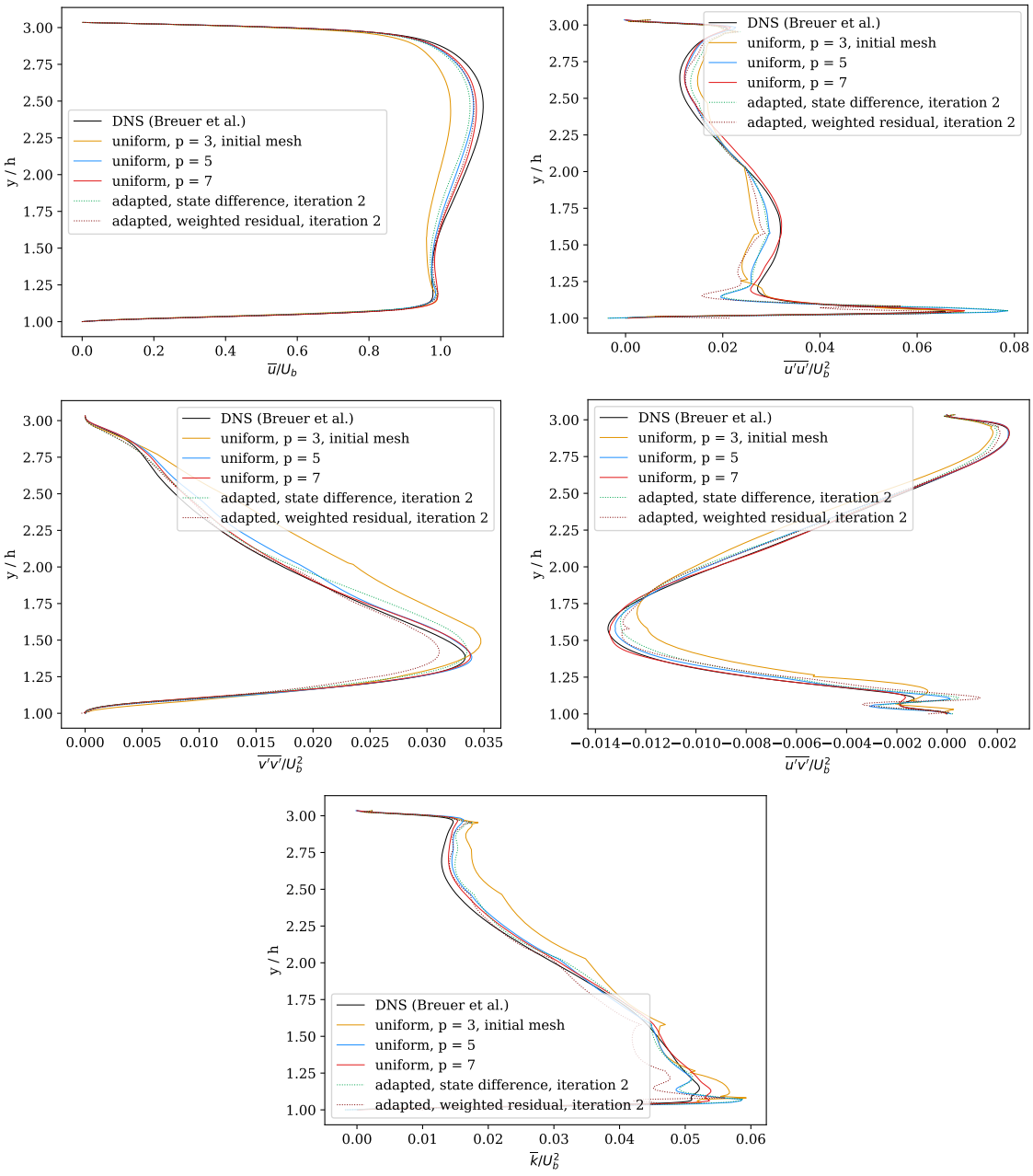


Figure 5.17: Various averaged statistical profiles relative to the DNS of Breuer *et al.* at $x/h = 0.05$.

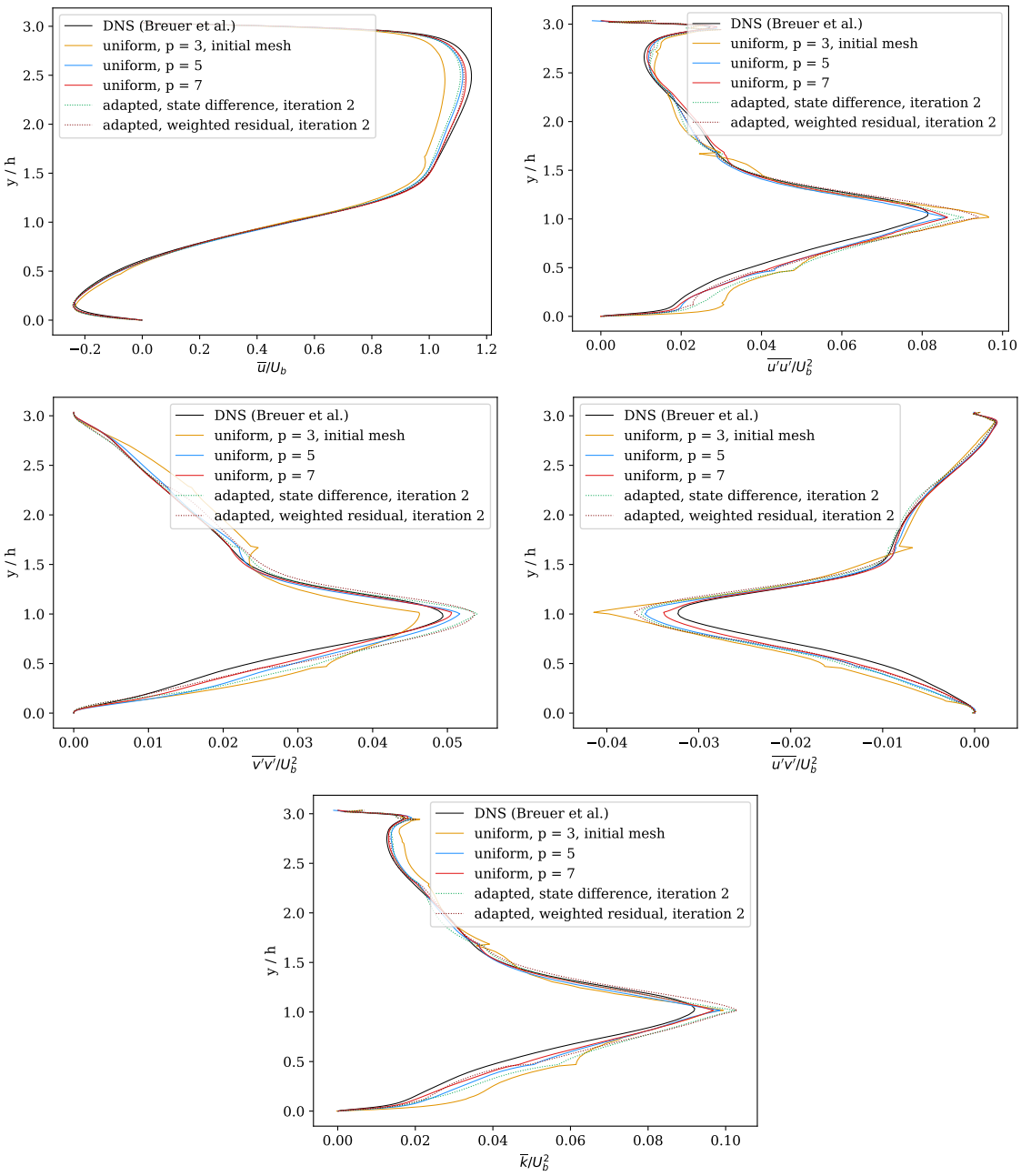


Figure 5.18: Various averaged statistical profiles relative to the DNS of Breuer *et al.* at $x/h = 2.0$.

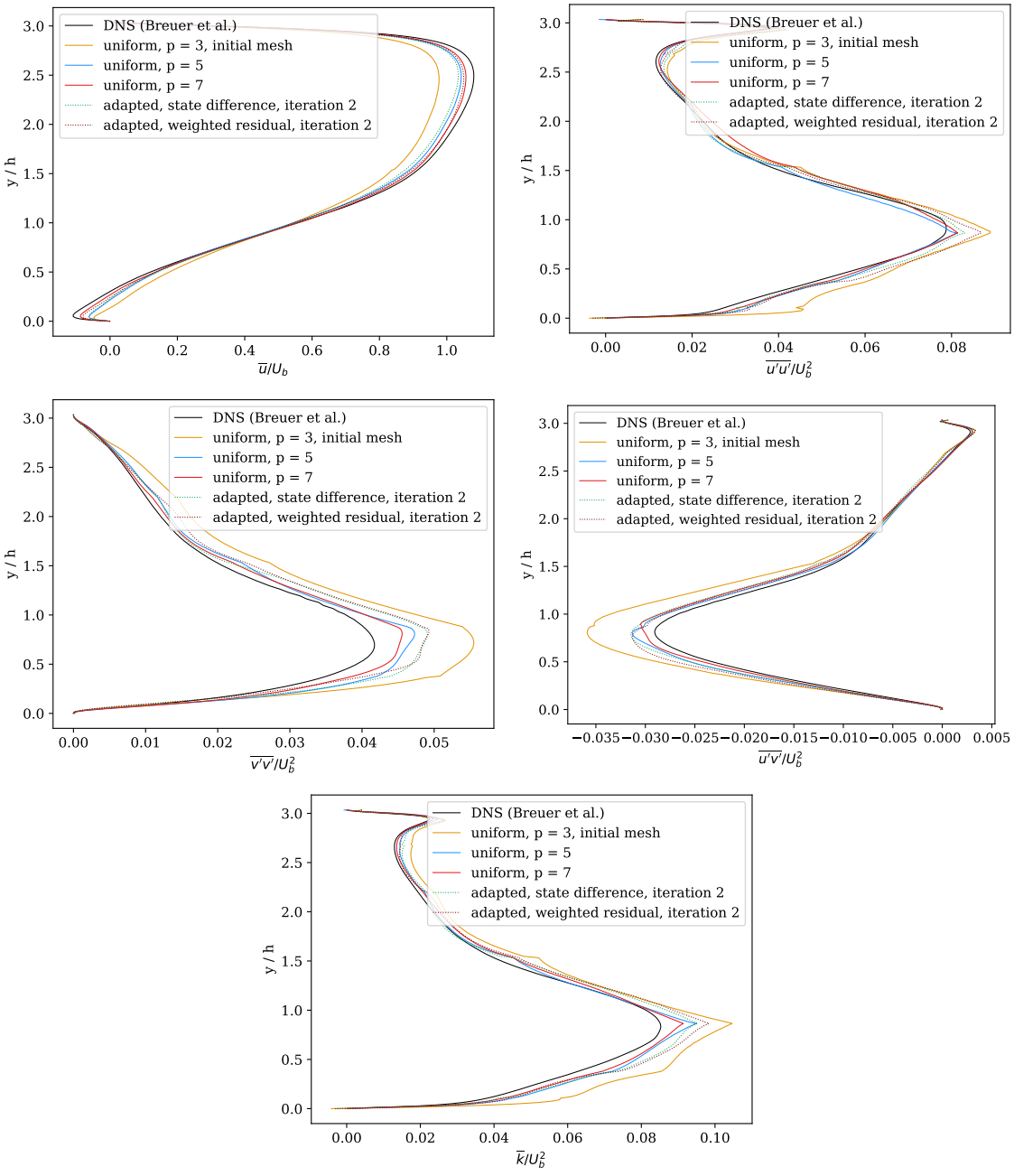


Figure 5.19: Various averaged statistical profiles relative to the DNS of Breuer *et al.* at $x/h = 4.0$.

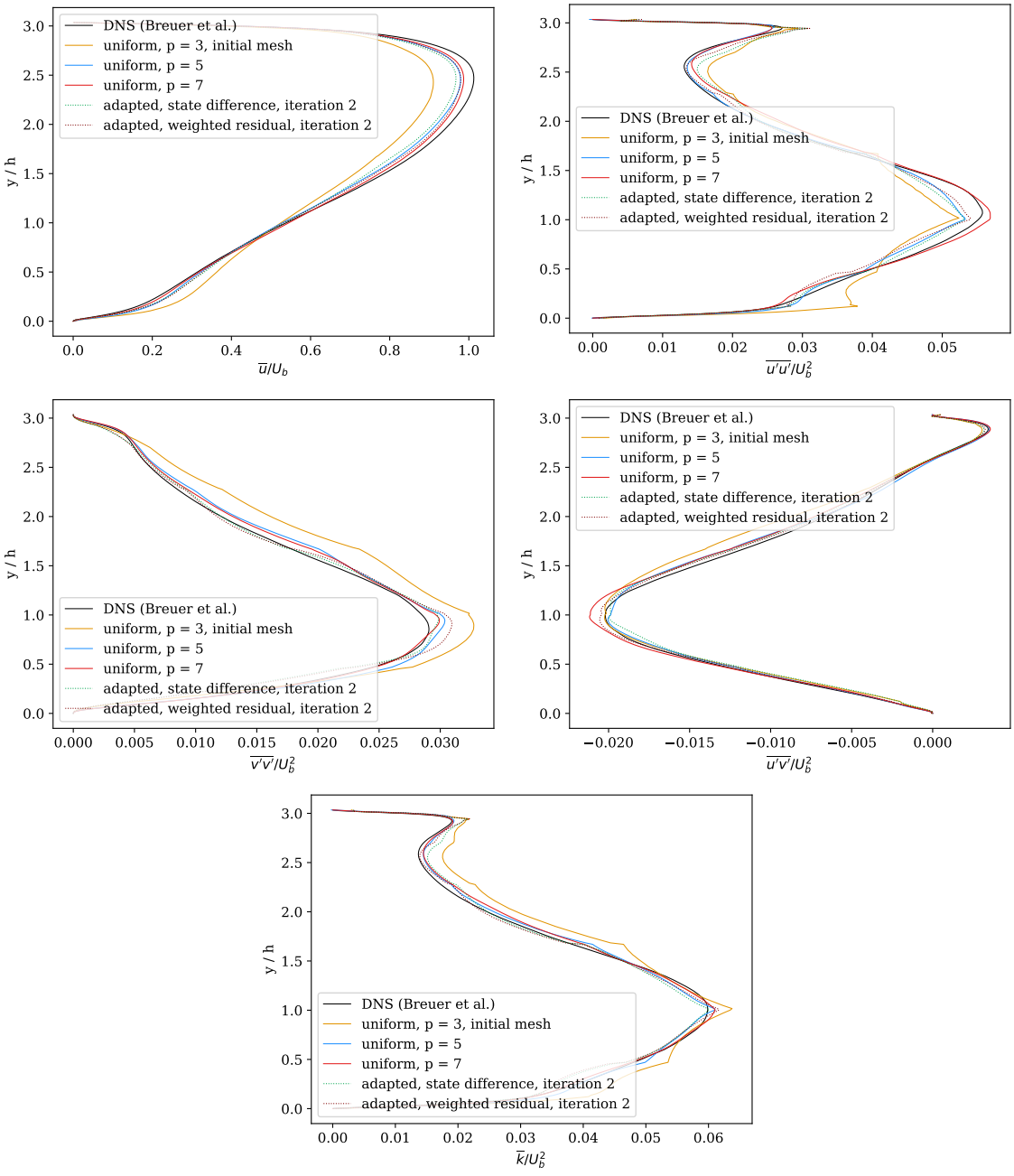


Figure 5.20: Various averaged statistical profiles relative to the DNS of Breuer *et al.* at $x/h = 7.0$.

about 40% fewer degrees of freedom. A look at Table 5.6 shows that the weighted residual indicator has used about 5,000 more degrees of freedom than the state difference indicator. This may explain some of the performance discrepancy.

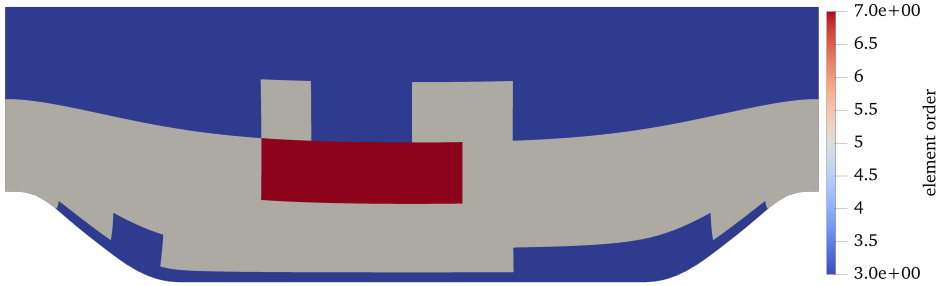
case	degrees of freedom
uniform $p = 3$	65,536
uniform $p = 5$	221,184
uniform $p = 7$	524,288
adapted, state difference, iteration 1	97,152
adapted, state difference, iteration 2	131,072
adapted, weighted residual, iteration 1	97,152
adapted, weighted residual, iteration 2	136,832

Table 5.6: Degrees of freedom for various $Re_b = 2800$ periodic hill cases.

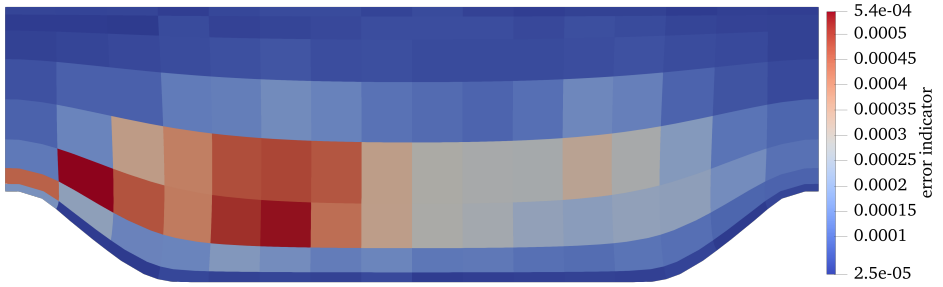
Moving on to the higher-order statistics seems to show a different story. Through most stations, the adaptation pattern produced by the state difference error indicator appears to outperform the weighted residual version. Looking closely, the difference is most clear near the bottom of the channel, but the top is not so clear. This difference is explained by the adaptation patterns shown in Figures 5.21 and 5.22. The difference indicator has focused more on lower parts of the channel in regions of high Reynolds stress. It is also apparent that the difference indicator used fewer high-order elements, preferring to spread the adaptation instead. The tendency to spread the adaptation pattern is made clear by the error indicator plots in Figure 5.21. The state difference indicator indicates significantly reduced error in the adapted region, causing the adapted region to expand to coarser regions on the next iteration. On the other hand, the weighted residual indicator does not show the same neat drop in indicated error. This is likely do to the noisy nature of the reconstruction, and the fact degrees of freedom are allowed to cancel when taking the difference between the fine and coarse space adjoints before taking the absolute value. This is the same issue observed above in the channel cases.

Figure 5.23 shows a skin friction comparison between uniform refinement and the second adaptation iteration for each error indicator. The "waves" in the skin friction profile are

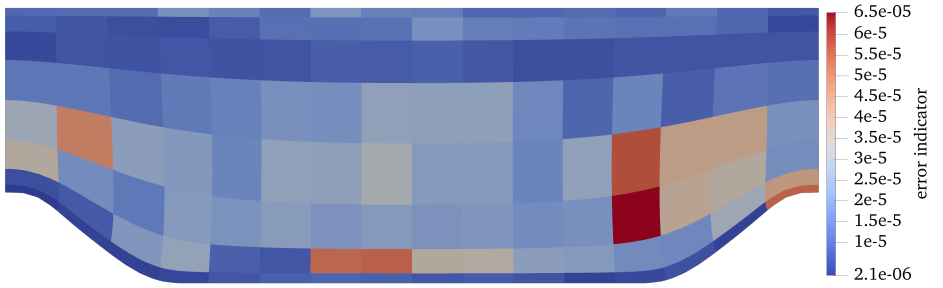
(a) Element orders, iteration 1



(b) Element orders, iteration 2



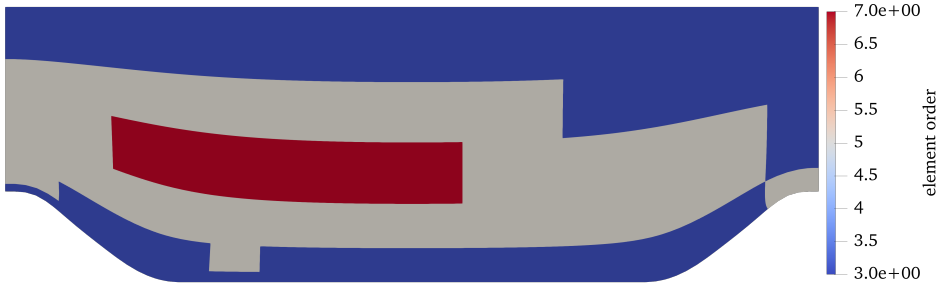
(c) Element error indicator, iteration 1



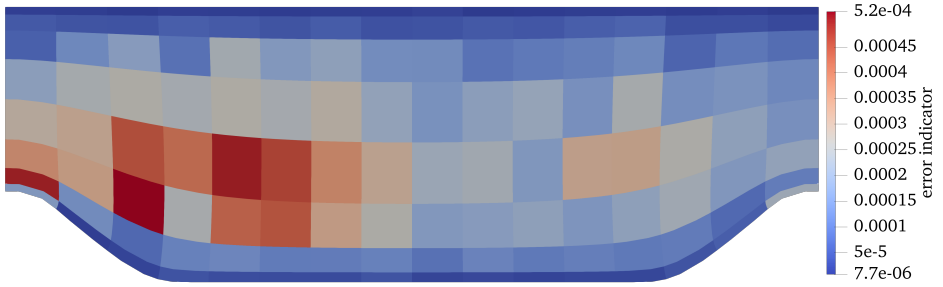
(d) Element error indicator, iteration 2

Figure 5.21: Element orders and indicated error for two adaptation iterations of the periodic hill using the state difference indicator.

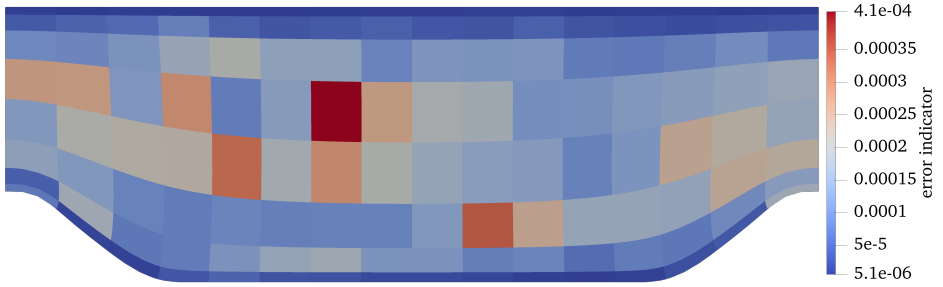
(a) Element orders, iteration 1



(b) Element orders, iteration 2



(c) Element error indicator, iteration 1



(d) Element error indicator, iteration 2

Figure 5.22: Element orders and indicated error for two adaptation iterations of the periodic hill using the adjoint weighted residual indicator.

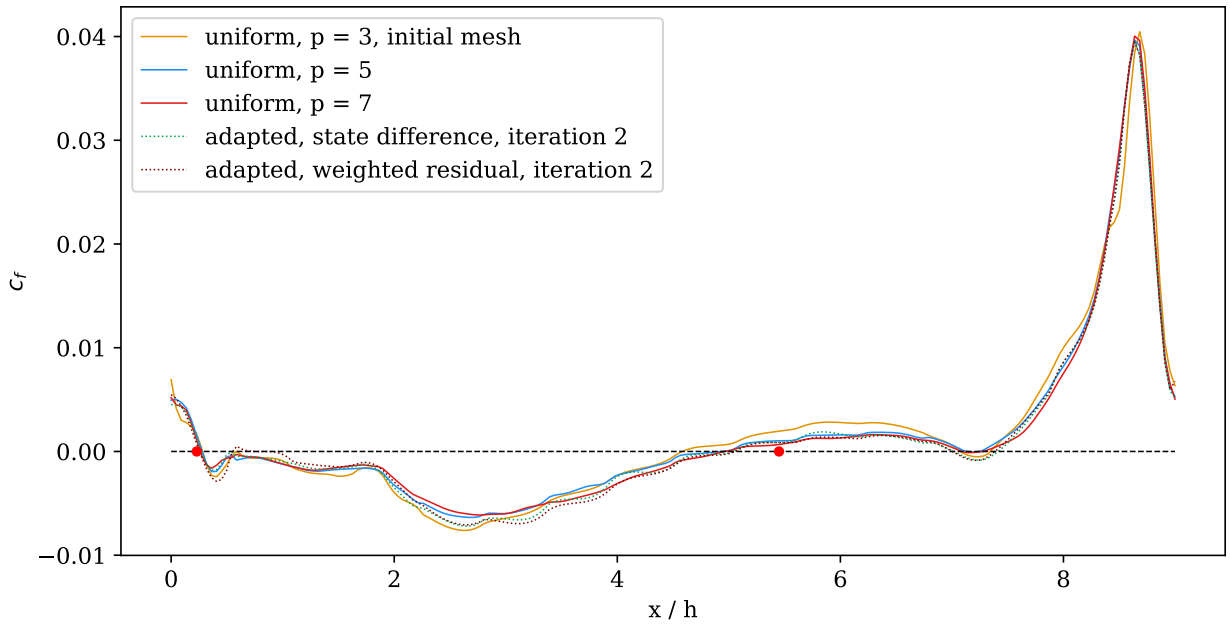


Figure 5.23: Skin friction coefficient comparison between uniformly refined and adapted periodic hills. DNS separation and reattachment points from Balakumar [7] are shown as dots.

likely element effects. They are consistent between cases, as expected, since each case is run on the same mesh. The adapted solutions predict the separation and reattachment points approximately as well as the $p = 7$ solution. Though, it is possible some of this is due to large element effects.

The error indicators on this case perform reasonably well, though perhaps only slightly better than uniform refinement. In that respect it is possible the performance has degraded moving to a more complex test case. The entropy-adjoint-weighted-residual error indicator does not show the same performance deficit that it showed on the unbiased channel test case. It did show some consistent themes though, such as relatively weak indicated error reduction from one iteration to the next. The state difference indicator continued to show strong reduction in indicated error. These differences could be due to the cancellation issues discussed in the channel section.

5.7 Trailing Edge Cooling Slot Test Case

5.7.1 Differences in Network Design

The trailing edge cooling slot test case used a previous version of the state difference error indicator. That indicator is described below.

For the model to generalize well on unseen data, a nondimensionalization process, or in the machine-learning perspective, a data pre-processing procedure, is applied before the network training. The resulting model has the form

$$\mathbf{U}_{h,c} = F_{sr}^{nn}\left(\frac{\mathbf{U}_{H,c,s} - \mathbf{u}_{m,s}}{\mathbf{u}_{\mathrm{rms},s}}, \frac{\mathbf{U}_{H,n,s} - \mathbf{u}_{m,s}}{\mathbf{u}_{\mathrm{rms},s}}, \log\left(\frac{V_{\mathrm{rms},c,d}\Delta_d}{\nu}\right)\right) u_{\mathrm{rms},s} + \mathbf{U}_{h,c}^{H} \quad \forall d \in \mathcal{D}, n \in \mathcal{N}, s \in \mathcal{S} \tag{5.15}$$

where the reconstructed state $\mathbf{U}_{h,c}$ is formed by adding the super-resolution model output to the prolonged coarse state $\mathbf{U}_{h,c}^{H}$, both on the central element c. $\mathcal{D}$ is the set of coordinate directions, $\mathcal{N}$ is the set of neighboring cells, and $\mathcal{S}$ is the full set of state ranks. $\mathbf{U}_{H,c,s}$ is the full state vector U restricted to element c and rank s. Likewise $\mathbf{U}_{H,n,s}$ is the same for each neighbor state. $\mathbf{u}_m$ and $\mathbf{u}_{\mathrm{rms}}$ are the mean and the root mean square of the state solution, and $V_{\mathrm{rms},c,d}$ is the root mean square velocity on element c in direction d. The last input feature in Equation 5.15 is the elemental Reynolds number, which is an indicator of the under-resolution present in the local element. Due to the wide range of the elemental Reynolds number, a logarithm is taken to help the model training.

In this work, we consider DG solutions on hexahedral meshes using tensor-product nodal basis functions. The network is then trained on three-dimensional rectangular meshes. A simplified two dimensional version of the proposed network structure for super-resolving one state component from $p = 1$ to $p = 3$ is sketched in Figure 5.24. A final input data permutation step generates additional samples by rotating a given training sample through all positive volume rotational symmetries of a cube to remove directional bias in the trained model.

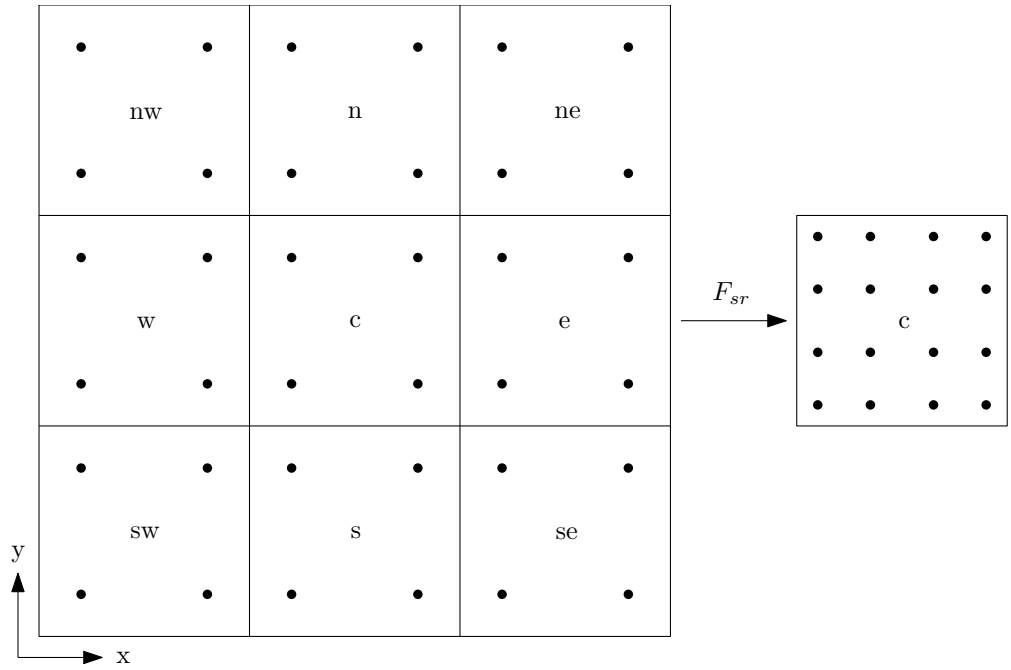


Figure 5.24: Simplified 2D super-resolution neural-network model from $p = 1$ to $p = 3$.

5.7.2 Error Indicator

Since the super-resolution model estimates the difference between a coarse state and a fine state, the magnitude can be used as an error indicator. Specifically, we have chosen to take the discrete L^1 norm of the super-resolution output on each element k.

$$e_k = \|F_{sr}^{nn}(\mathbf{U}_{H,k})\mathbf{U}_{\text{rms},k}\|_1 \tag{5.16}$$

The element-wise indicator allows localized element adaptation. The per-element indicator may be averaged over statistically constant regions, such as wall normal layers in a turbulent channel, to reduce noise in the indicated error. We also average the error indicator over many snapshots to further reduce noise for statistically steady flows. This is a fairly straightforward error indicator that will simply show error where under-resolution is present but not necessarily indicate the cause.

5.7.3 Data Generation and Network Training

Training data for the super-resolution neural-network are generated by capabilities we have added to NASA's *eddy* code[31, 33]. All training and test cases use hexagonal elements with tensor-product basis functions. For super-resolution, the network input neighborhood consists of the central element of interest and all elements directly across a face for a total of seven elements. The length scale used for element Reynolds number generation is the length of the axis-aligned bounding box in each dimension. The training data are generated by projecting cases simulated at high-order down to the relatively low-orders required for super-resolution. We have generated training data by sampling an $Re_\tau = 950$ turbulent channel case. Once we have adapted an initial constant order solution there will be multiple solution orders in the same domain. This situation is handled by training a separate neural-network for each potential order. When training data are generated, neighboring elements are always projected down to a coarse order based on the solution order of the central element. We use the same core code for training and adaptation, the only difference being when training, the input data are generated by projecting a high-order solution to low-order and when online, the input data to the network are generated by the present simulation state.

5.7.4 Cooling Slot Test Case

To demonstrate our method, we have chosen a turbulent mixing test case meant to mimic a cooling slot in turbomachinery, presented in [54]. This case uses auxiliary domains to produce a turbulent boundary condition at the inlet. The upper auxiliary domain is a one-way coupled periodic boundary layer simulation where boundary layer thickness is maintained by a specialized forcing field. Source terms are added to the Navier-Stokes equations corresponding to the mean gradient of each state according to a log-law and wake profile. The lower auxiliary domain is a one-way coupled turbulent channel driven by a simple constant body force. The lower auxiliary domain connects directly to the cooling slot of height y_c.

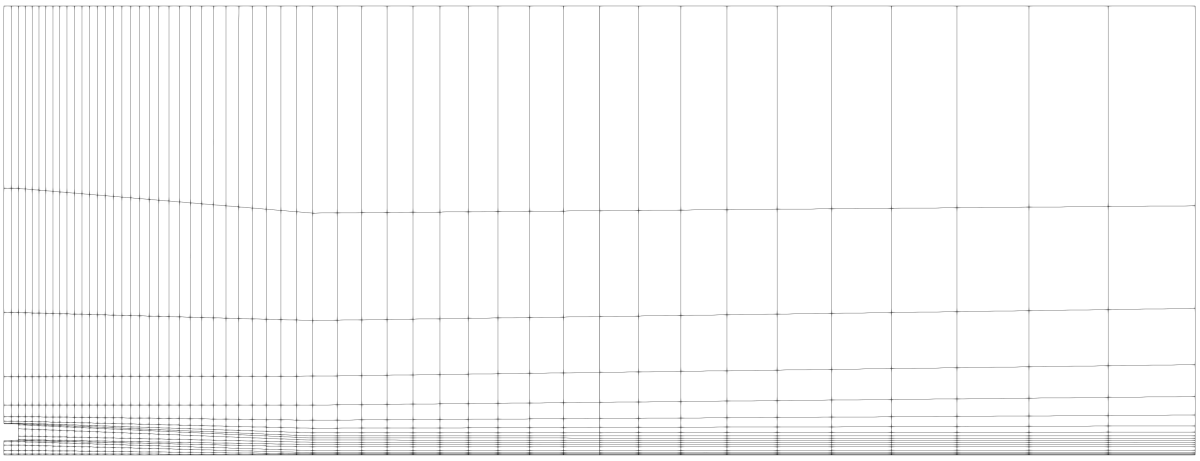


Figure 5.25: Element boundaries for the primary domain of the trailing edge cooling slot case. Turbulent inflow auxiliary domains are to the left, perfectly matched layer outflow domains are to the top and right.

Inflow from the auxiliary domains is initially separated by a small lip in the main computational domain before mixing, as shown in the lower left of each domain in Figures 5.25 and 5.26. The end of the lip is positioned at $x/y_c = 0$.

Each adaptation iteration refines approximately 20% of elements. Since the problem is statistically homogeneous in the spanwise direction by construction, elements sharing spanwise faces are grouped and adaptation is performed in an effectively two dimensional space with these element groups. Only elements with a downstream distance less than approximately $x/y_c = 32$ are adapted to ensure there is no excessive downstream refinement.

The flow-through time scale for the main computational domain is $t = 80y_c/U_\infty$. In all cases, averaging is started after waiting at least $50.6t$ from the initial condition and averaging takes place for at least $10.1t$, all at a time step of approximately $5 \times 10^{-4}t$. This integration time exceeds that of [54], which used $2 - 3t$. This ensures that the difference between the profile at half averaging time and full averaging time is small relative to the difference between profiles for different simulations. The profile at each station is generated from the average of 20 spanwise velocity samples and all simulations of this case use the same high resolution

(p = 7) inlet regions.

5.7.5 Results

The adapated order distribution after two iterations next to a reference snapshot is shown in Figure 5.26. Adaptation has focused in the wake region of the blunt body separating the channel and boundary layer inflows. The adaptation pattern appears strongest in the region of the most intense vortex shedding. The region immediately after the blunt body has little adaptation save for a few elements near its corners. Lower orders are also present further downstream, indicating they were not the regions of maximum indicated error. While the most complex vortex shedding region has been highlighted by the indicator, some key flow features could probably use more adaptation. The corners of the blunt body off of which the vortices are shedding should be influential in the down stream shedding effect. They are captured by one adaptive iteration, but are not raised to the maximum order possible, $p = 7$.

Table 5.7: Slot case degree of freedom counts in primary computational domain. The adapted result has undergone two adaptive iterations.

p = 3	adapted	p = 5	p = 7
592,640	1,313,600	2,000,160	4,741,120

Velocity profiles at several stations are showin in Figure 5.27. Where available, these profiles have comparisons with much higher resolution results from [54]. We see that the adpated result performs slightly worse at each station than the uniform $p = 5$ solution. The adpated solution uses approximately 34% fewer degrees of freedom than the $p = 5$ solution as shown in Table 5.7. This places the performance of the adapted indicator roughly in line with uniform refinement. The result is consistent with observations from the periodic hill case earlier.

Higher order statistical profiles are provided at various stations in Figures 5.28, 5.29, 5.30, 5.31, 5.32. These figures also include comparison with Garai *et al.* [54] where available.

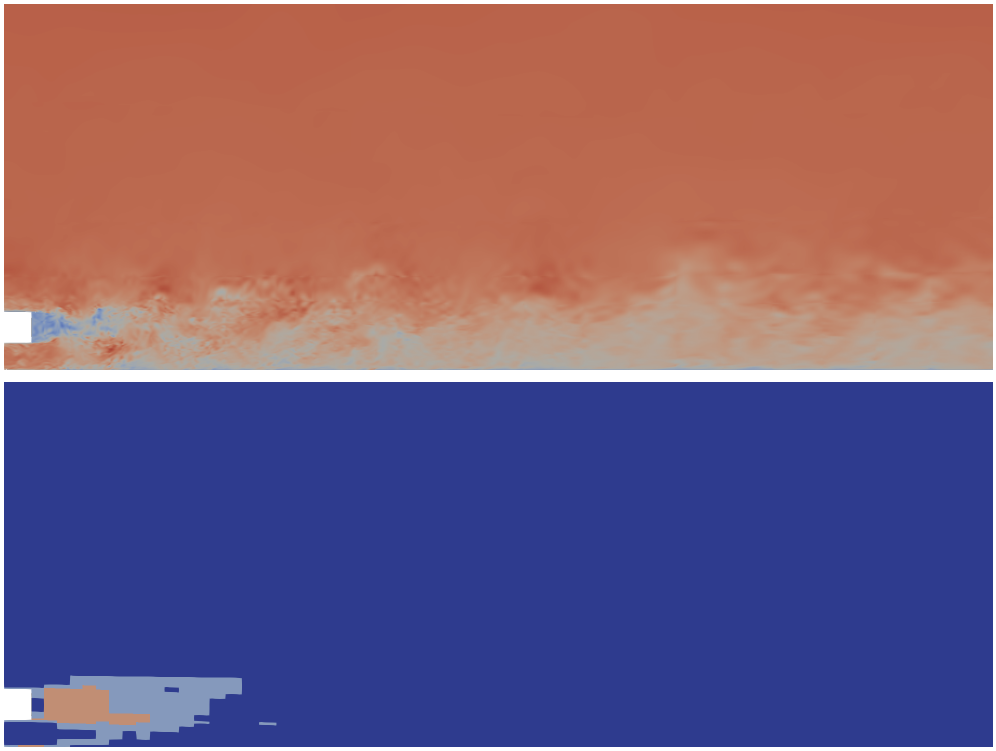


Figure 5.26: Mixed $p = 3, 5, 7$ adapted order distribution and reference velocity field.

Overall, despite lacking the $p = 5$ data on these plots, the story remains roughly consistent. The adaptive method appears to perform roughly consistently with uniform refinement on large test cases. Specific stations are of note. Reynolds shear stress variation at $x/y_c = 10$ is extreme. This is because this station is near the center of the heavy vortex region. We also saw in Figure 5.26 that this is the most heavily adapted region. Therefore, the extreme shear stress profile is well resolved by the adapted solution, at least relative to uniform refinement. Even the $p = 7$ solution is significantly off the true solution, but the case is still under-resolved at that resolution.

5.8 Summary

In this chapter mesh adaptations have been performed using various super-resolution based error indicators on turbulent flow problems. A 20% fixed fraction adaptation strategy was employed. We began with a channel case with no initial mesh refinement. This is meant

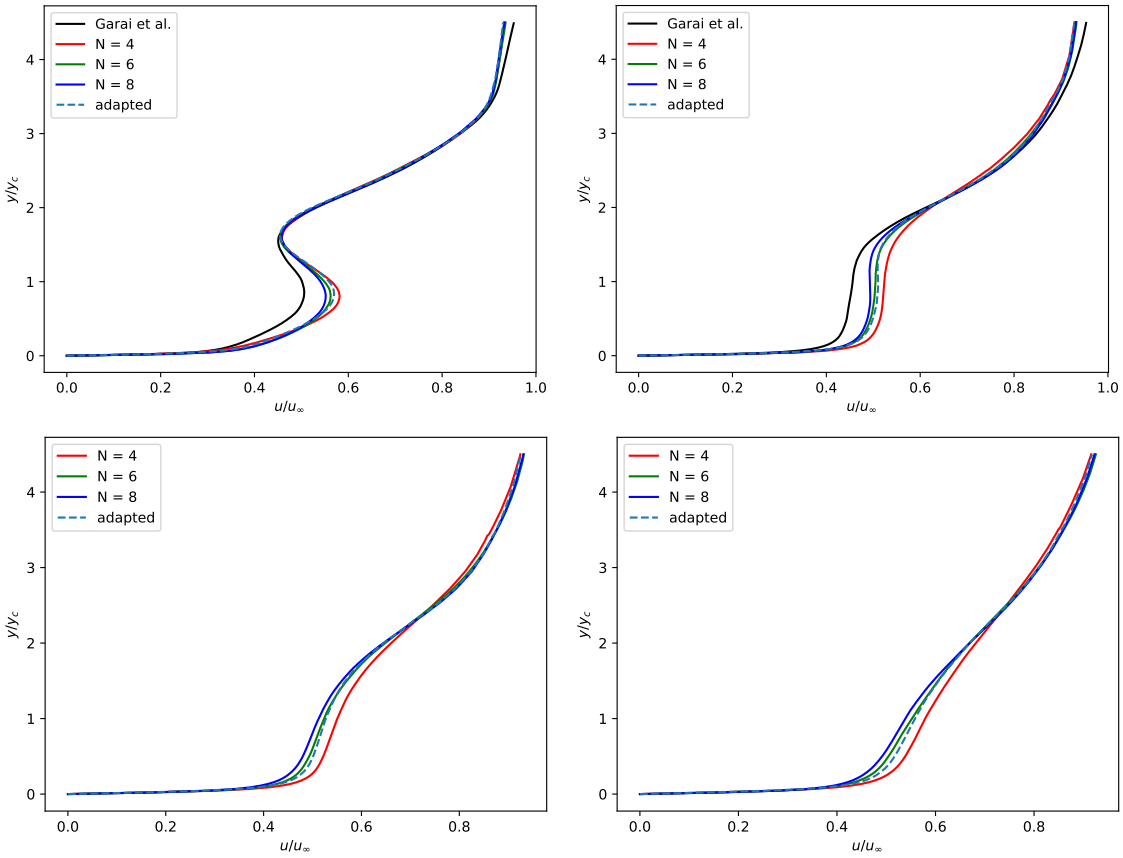


Figure 5.27: Normalized velocity profiles for the slot case at various downstream stations. From top left to bottom right the stations are $x/y_c = 4$, $x/y_c = 10$, $x/y_c = 20$ and $x/y_c = 30$.

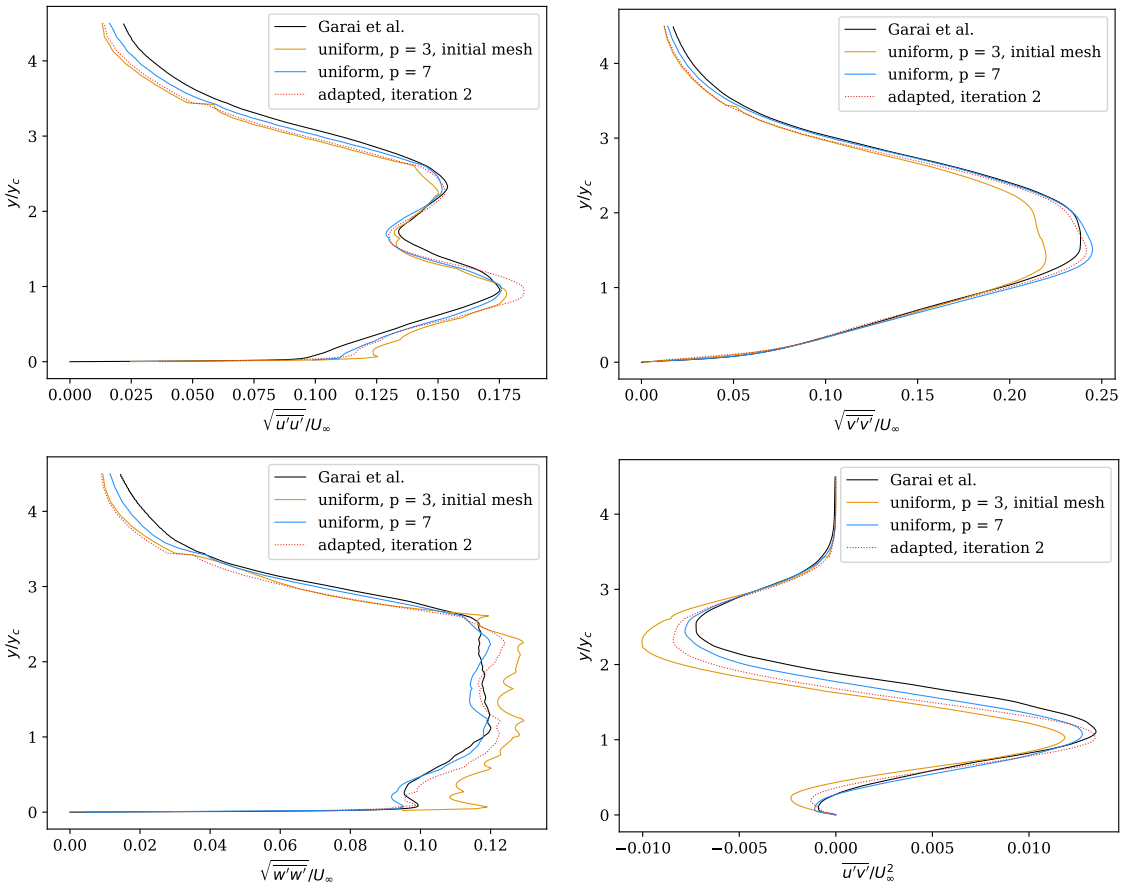


Figure 5.28: High-order slot case statistics compared with uniform refinement at $x/y_c = 4$.

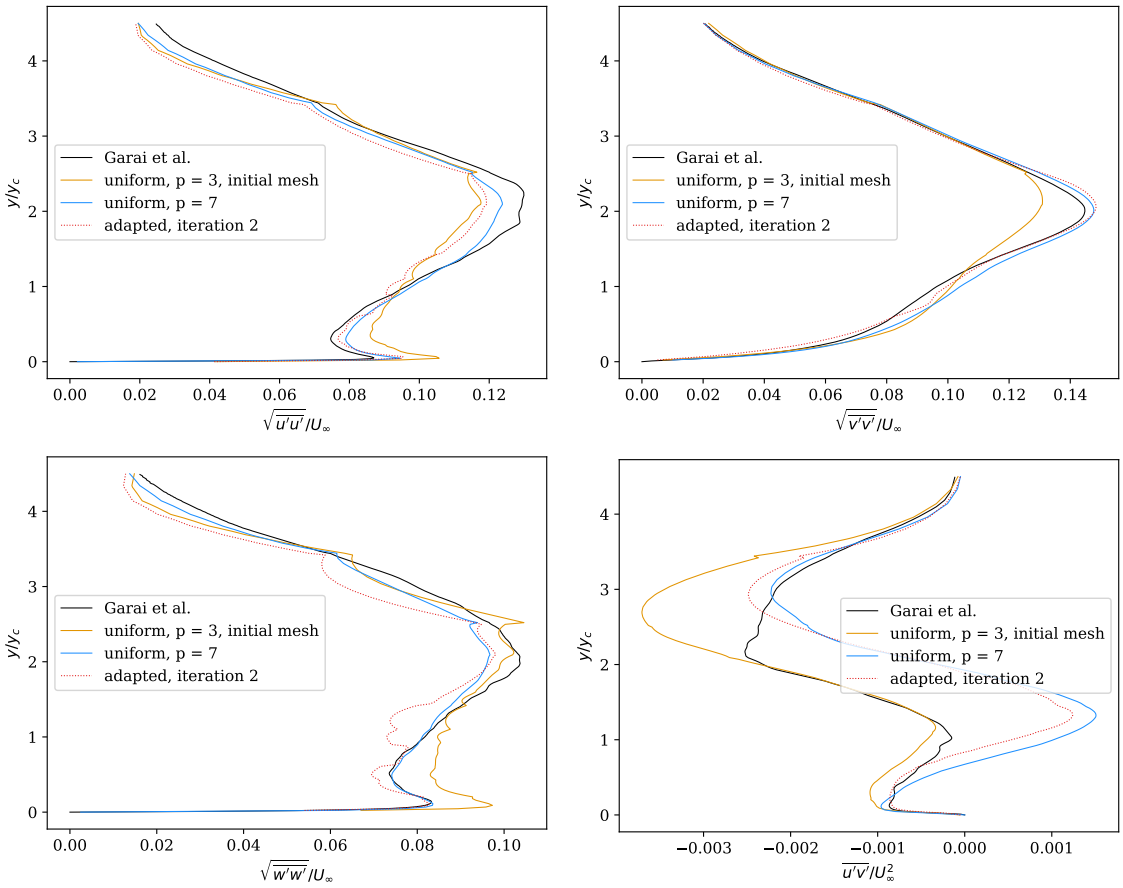


Figure 5.29: High-order slot case statistics compared with uniform refinement at $x/y_c = 10$.

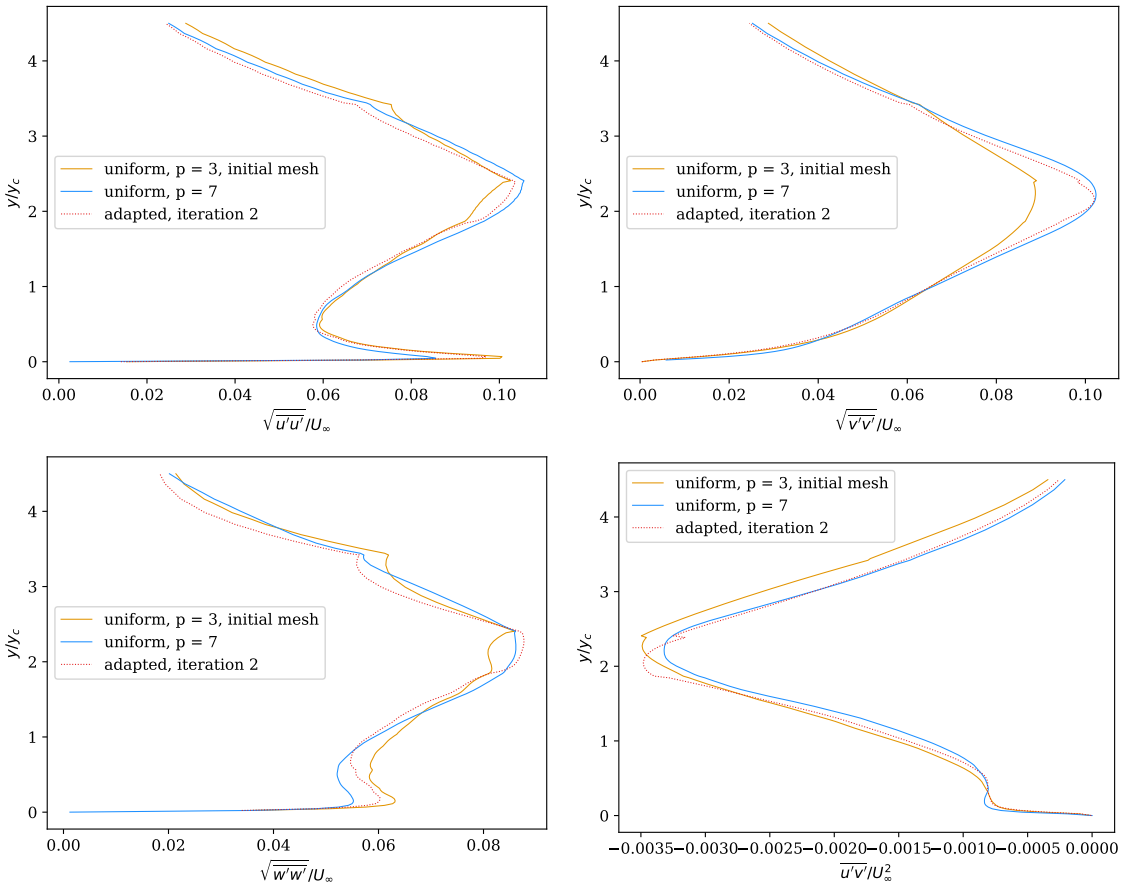


Figure 5.30: High-order slot case statistics compared with uniform refinement at $x/y_c = 20$.

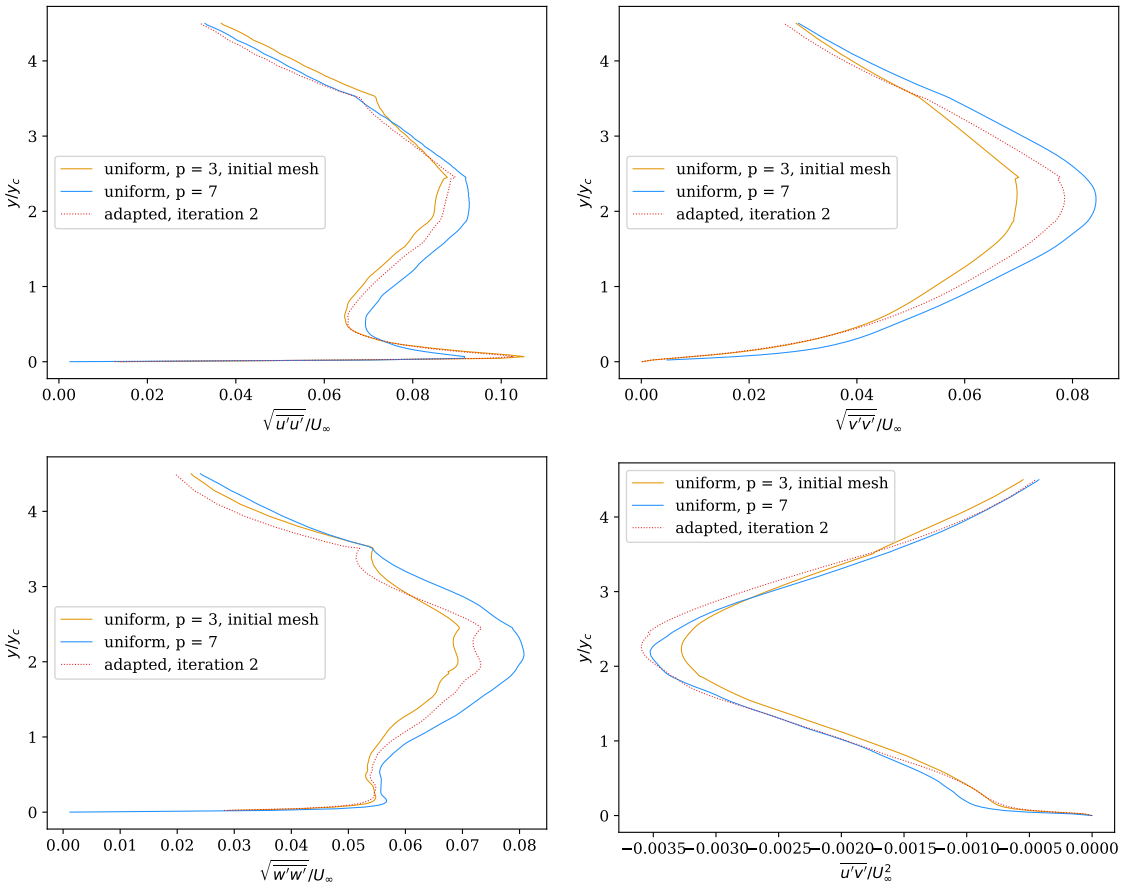


Figure 5.31: High-order slot case statistics compared with uniform refinement at $x/y_c = 30$.

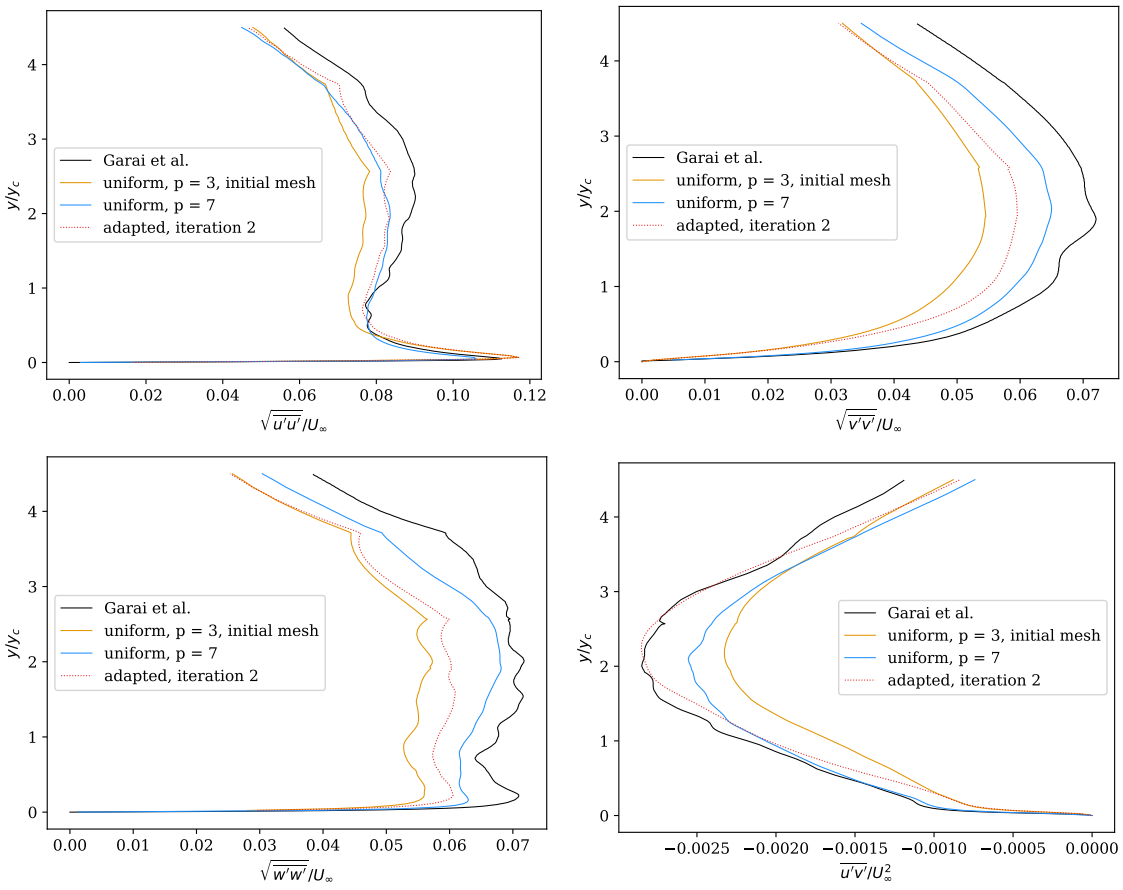


Figure 5.32: High-order slot case statistics compared with uniform refinement at $x/y_c = 50$.

to promote severe under-resolution in the near-wall region. The state difference indicator returns favorable results on this case. It always adapts the near-wall region, while simultaneously adapting farther from the wall when necessary. The weighted residual error indicator performs poorly on this case, missing the first element off the wall at low Reynolds numbers. This error improves at higher Reynolds numbers, where the near-wall region is captured. The poor performance of the weighted residual version could be due to the averaging behavior. While mathematically sound, averaging the resulting indicators from the entropy adjoint weighted residual is suboptimal relative to averaging the inputs [10]. The state difference indicator was also compared against a velocity-gradient-based indicator using the same adaptation strategy. The tendency of the super-resolution-based indicator to expand the adapted region is encouraging. Overall results are similar between the two indicators. It is possible the state difference indicator should be thought of as a feature-based error indicator that indicates where gradients are high.

Moving on to problems with initially refined meshes shows less favorable results. Testing on the periodic hill shows similar results between the error indicators. These results are each more in line with uniform refinement than previous results on the channel case. Results appear to deteriorate on more complex meshes with reasonable initial refinement patterns. This trend continues on the trailing edge cooling slot case, where a modified version of the state difference indicator is tested. The initial mesh has heavy refinement in the most interesting flow region downstream of the cooling slot. Results are once again in line with the uniformly refined variants. It appears that, overall, the technique is successful for a sufficiently poor initial mesh. Super-resolution-based indicators are unable to hone in on fine flow features that control the flow, such as separation points.

Chapter 6

Conclusions

In this chapter we will present a summary of the work performed in this thesis and discuss the contributions. Suggestions are made for impactful research directions in turbulent adatpation. Finally, future work for the improvement of super-resolution-based indicators is discussed.

6.1 Summary

On a sufficiently poor initial mesh, super-resolution-based error indicators can perform quite well. We saw this with the state difference indicator on the channel flow problem. Moving to the more complex test cases, performance deteriorated until it was in line with, or only slightly better than, uniform refinement. The more complex test cases have reasonable initial mesh refinement patterns. The periodic hill mesh is refined near the wall, and the cooling slot mesh is refined in the wake of the slot. With these initial refinements, the indicators are asked to move from predicting regions of general activity, to specific regions relevant to the flow. We have seen that, at this point, the indicators fail to focus refinement further. The periodic hill and cooling slot cases both feature flow separation. Capturing the separation point and the relevant upstream regions should be crucial for accurate flow prediction. However, neither

error indicator is able to do this. This is not surprising, considering the error indicators have no mechanism to derive such information from the flow. These results reinforce the need for truly output-based indicators.

In Chapter 3, a high-order KS equation solver was used to test the strengths and limitations of the chosen network architecture. It was shown that the network is capable of nearly perfect reconstruction of unseen data in the same mesh and of the same equation parameterization. Checking if training on one parameterization generalized to another, we found that it does not. This lack of generalization does not depend on the frequencies of the training and test states. Both training on low-wavenumber data with testing on high-wavenumber data and training on high-wavenumber data with testing on low-wavenumber data result in poor generalization. Mixing training data with the same network recovers most of the performance of the individual training sets, but allows generalization. Some bias was observed toward the mean of the training sets during this testing.

Chapter 4 continues the work of Chapter 3 in higher dimensions. It is shown that the baseline reconstruction of velocities on two dimensional finite-element states is of high quality. The quality can be further improved by modifying the network architecture to perform a series of simpler reconstruction problems. The importance of controlling inter-element discontinuities is emphasized. The discontinuities can be observed in energy spectra as excess energy in high-wavenumber regions. We continue on to experiment with the quality of reconstruction in three dimensions. The simple fully-connected network architecture is carried forward for testing in 3D. We saw the reconstruction quality degrade further, but significant spectral correction remains present. A variety of network sizes were tested, leading to a final, slightly over-sized, network choice. The generalizability of this network across turbulent channel Reynolds numbers was then tested, similar to the tests in Chapter 3. A network trained on a mixed data set showed similar reconstruction performance to networks tailored to individual data sets. This single mixed network was chosen for use in the adaptations in Chapter 5.

Chapter 5 presents two error indicators based on super-resolution reconstruction. Those indicators were then used on several test cases ranging from sanity-check to large-scale. We began with the turbulent channel test cases, where the mesh was not refined near the wall. This allowed the adaptive indicators to demonstrate their "preferred" element order distribution. The simple state difference indicator performed well on this test. It showed a smooth initial error indication over the channel, with appropriate adjustments after the first iteration of refinement. The indicated error also significantly decreased overall, as expected, without ignoring the under-resolved wall. In comparison, the performance of the weighted residual indicator was puzzling. The indicated error, while biased toward the walls, was significantly noisier than the state difference version. This led to failure to adapt the first element off the wall at the lowest Reynolds number and significantly degraded performance as a result. This problem decreased somewhat at higher Reynolds numbers. One possible culprit for this behavior is cancellation in the adjoint calculation. Another is averaging the error indicator itself instead of the adjoints and residual separately.

Chapter 5 continued with the periodic hill test case. This test is a canonical case of flow separation and recirculation, making it an ideal case for our testing. The difference between the indicators was less stark with this case. Similar performance was achieved on both error indicators, with the weighted residual indicator favoring the top of the domain and the difference indicator favoring the bottom. Each network performed roughly in line with, or slightly favorably, relative to uniform refinement on a degree of freedom basis. The trailing edge cooling slot test case used a slightly older version of the state difference error indicator. Performance of this indicator is consistent with the current version on the uniformly refined channel test case. On a variety of turbulence statistics, the adapted mesh performs roughly similarly to the periodic hill case, in line with uniform refinement.

Performance of these new methods is encouraging on simple test cases, but performance appears to degrade as the cases become more complex. A variety of possible improvements

are discussed below.

6.2 Contributions

The major contributions of this dissertation are:

1. **Introduced error indicators based on super-resolution reconstruction.** The simple state difference indicator tended to perform better than the entropy adjoint weighted residual version. This could be because the chosen averaging technique was suboptimal for the weighted-residual indicator.

2. **Demonstrated that a single reconstruction model trained on a variety of flows is sufficient for adaptation.** Reconstruction performance from a single model on various flows matched or exceeded the performance seen from more specially trained models.

3. **Implemented reconstruction-based adaptation in a high-order discontinuous Galerkin code.** The adaptive framework incorporates super-resolution reconstruction by directly modifying an existing DG code.

4. **Performed super-resolution-based adaptation on several chaotic flow problems.** The turbulent channel tested the ability of the super-resolution-based error indicators to refine an extremely poor initial mesh. The periodic hill test case demonstrated their performance for separated flows. The trailing edge cooling slot provided another flow separation test case at larger scale.

6.3 Research Outlook

In engineering, we are interested in computing time-averaged outputs, *e.g.* drag, from turbulent flows. We are also interested in safety-critical phenomena like flow separation. We

would like our mesh adaptation methods to capture these phenomena to facilitate the design process by reducing cost for a given accuracy. We may break mesh adaptive techniques into a couple major forms. The first uses features of the flow to refine the mesh. These features may be Mach number jumps, pressure gradients, vorticity, and many more. Adapting on these features can lead to suboptimal results, since they do not directly capture outputs of interest. The second form of adaptive technique measures the sensitivity of the output of interest, like drag, to the discretized equations. This form finds an optimal mesh for a given output, one that computes the output at the highest possible accuracy for a fixed cost. Super-resolution-based indicators fall in the first category. On a sufficiently poor initial mesh, a reconstruction-based indicator is able to outperform uniform refinement. The indicator will show where there is activity in the flow, and where there is not. The indicators are not able to concentrate on detailed flow features within a turbulent regime.

To solve this problem, and turbulent adaptation generally, indicators using super-resolution reconstruction are not sufficient. A viable method for chaotic sensitivity analysis is needed instead. Least squares shadowing has proven able to perform chaotic sensitivity analysis, albeit at high computational cost. The cost has come down over time with the development of methods like non-intrusive least squares shadowing. Additional research effort in this general direction may yield an output-based indicator for chaotic flows at practical computational cost.

It appears that truly output-based indicators will be needed for large, *e.g.* orders of magnitude, efficiency gains in the simulation of turbulent flows. If this is the case, another useful research direction to pursue in the meantime is the efficiency of solvers themselves. Modern computer hardware tends to be rich in compute performance and relatively poor in memory bandwidth. This means that only algorithms that require a large number of floating point operations per memory access are able to fully use the available hardware. These algorithms are said to be compute-bound. In contrast, relatively low-order or non-compact schemes, like

finite-volume, tend to be memory-bound. That is, their speed is limited by the processor's ability to access memory, not the speed of the processor itself. High-order, compact stencil finite element methods provide a possible route to compute-bound solvers on modern architectures. The computation in these methods boils down to matrix-matrix multiplication and similar operations. Matrix-matrix multiplication is a known compute-bound problem, if the matrices are sufficiently large. This means that finite element methods are candidates to take full advantage of modern hardware when run at high-orders. Research in this direction will also require work on nonlinear solvers to deal with the stiffness of high-order finite element methods.

6.4 Future Work for Super-Resolution-Based Indicators

- **Experiment with Weighted Residual Indicator Averaging**

  The error indicator averaging technique is mathematically sound, but better techniques could be used in practice. Averaging the reconstructed adjoint could expose consistent directional bias within an element and prevent the possible adjoint cancellation and additional noise. Ideally this averaging could be used to avoid reconstructing the state at each snapshot, saving computational resources. Perhaps a network could be generated to reconstruct an averaged state instead of an instantaneous state. As it stands however, the instantaneous snapshots would need to be computed.

- **Improve Super-Resolution Reconstruction Architectures for DG**

  Chapter 4 showed that improved reconstruction architectures are possible. More accurate reconstruction could help the accuracy of the error indicators and therefore the quality of adaptaion. DG basis function coefficients are a type of input distinct from most machine-learning literature on super-resolution, which usually deals with pixel data. This could mean that some novel architectures become interesting for this

purpose.

- **Implement Remeshing Techniques**

  p-adaptation is a fairly restrictive form of adaptation. The mesh cannot change, only the order with which the elements are represented. This could mean that small but significant flow features are missed by the adaptation. Anisotropic adaptation will be critical to capture important features of turbulent flows. Combining anisotropic mesh adaptation with order adaptation will yield the best results.

- **Improve 1D Testing**

  The 1D testing on the KS equation could be more extensive than at present. With support for variable element sizes and orders, contrived adaptation patterns could be set up with exact solutions. This could also be a test bed for more sophisticated adaptation strategies where an optimal mesh is found for a given cost, likely under some other restriction to prevent the obvious uniform distribution.

Bibliography

[1] Antonella Abbà, Alessandro Recanati, Matteo Tugnoli, and Luca Bonaventura. Dynamical p-adaptivity for LES of compressible flows in a high order DG framework. *Journal of Computational Physics*, 420:109720, 2020.

[2] Roger Alexander. Diagonally implicit Runge–Kutta methods for stiff ODE's. *SIAM Journal on Numerical Analysis*, 14(6):1006–1021, 1977.

[3] GP Almeida, DFG Durao, and MV Heitor. Wake flows behind two-dimensional model hills. *Experimental Thermal and Fluid Science*, 7(1):87–101, 1993.

[4] Saeed Anwar, Salman Khan, and Nick Barnes. A deep journey into super-resolution: A survey. *ACM Computing Surveys (CSUR)*, 53(3):1–34, 2020.

[5] Douglas N Arnold, Franco Brezzi, Bernardo Cockburn, and L Donatella Marini. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM journal on numerical analysis*, 39(5):1749–1779, 2002.

[6] Timothy J Baker. Mesh adaptation strategies for problems in fluid dynamics. *Finite Elements in Analysis and Design*, 25(3-4):243–273, 1997.

[7] Ponnampalam Balakumar and George Ilhwan Park. DNS/LES simulations of separated flows at high Reynolds numbers. In *45th AIAA Fluid Dynamics Conference*, page 2783, 2015.

[8] F Bassi, A Colombo, A Crivellini, M Franciolini, A Ghidoni, G Manzinali, and Gianmaria Noventa. Under-Resolved Simulation of Turbulent Flows Using a p-adaptive Discontinuous Galerkin Method. In *iTi Conference on Turbulence*, pages 157–162. Springer, 2018.

[9] Francesco Bassi, L Botti, Alessandro Colombo, Andrea Crivellini, Antonio Ghidoni, and F Massa. On the development of an implicit high-order discontinuous Galerkin method for DNS and implicit LES of turbulent flows. *European Journal of Mechanics-B/Fluids*, 55:367–379, 2016.

[10] Francesco Bassi, Alessandro Colombo, Andrea Crivellini, Krzysztof J Fidkowski, Matteo Franciolini, Antonio Ghidoni, and Gianmaria Noventa. Entropy-adjoint p-adaptive discontinuous Galerkin method for the under-resolved simulation of turbulent flows. *AIAA Journal*, 58(9):3963–3977, 2020.

[11] Andrea D Beck, Thomas Bolemann, David Flad, Hannes Frank, Gregor J Gassner, Florian Hindenlang, and Claus-Dieter Munz. High-order discontinuous Galerkin spectral element methods for transitional and turbulent flow simulations. *International Journal for Numerical Methods in Fluids*, 76(8):522–548, 2014.

[12] Marsha J Berger and Antony Jameson. Automatic adaptive grid refinement for the Euler equations. *AIAA journal*, 23(4):561–568, 1985.

[13] Patrick J Blonigan. Adjoint sensitivity analysis of chaotic dynamical systems with non-intrusive least squares shadowing. *Journal of Computational Physics*, 348:803–826, 2017.

[14] Patrick J Blonigan, Pablo Fernandez, Scott M Murman, Qiqi Wang, Georgios Rigas, and Luca Magri. Toward a chaotic adjoint for LES. *arXiv preprint arXiv:1702.06809*, 2017.

[15] Patrick J Blonigan, Steven A Gomez, and Qiqi Wang. Least squares shadowing for

sensitivity analysis of turbulent fluid flows. In *52nd Aerospace Sciences Meeting*, page 1426, 2014.

[16] Patrick J Blonigan, Qiqi Wang, Eric J Nielsen, and Boris Diskin. Least-squares shadowing sensitivity analysis of chaotic flow around a two-dimensional airfoil. *AIAA Journal*, 56(2):658–672, 2018.

[17] Jan Bohn and Michael Feischl. Recurrent neural networks as optimal mesh refinement strategies. *Computers & Mathematics with Applications*, 97:61–76, 2021.

[18] Michael Breuer, Nikolaus Peller, Ch Rapp, and Michael Manhart. Flow over periodic hills–numerical and experimental study in a wide range of Reynolds numbers. *Computers & Fluids*, 38(2):433–457, 2009.

[19] Peter N Brown. A local convergence theory for combined inexact-Newton/finite-difference projection methods. *SIAM Journal on Numerical Analysis*, 24(2):407–434, 1987.

[20] Peter N Brown and Youcef Saad. Hybrid Krylov methods for nonlinear systems of equations. *SIAM Journal on Scientific and Statistical Computing*, 11(3):450–481, 1990.

[21] Philip Claude Caplan, Robert Haimes, David L Darmofal, and Marshall C Galbraith. Four-dimensional anisotropic mesh adaptation. *Computer-Aided Design*, 129:102915, 2020.

[22] Philip Claude Delhaye Caplan. *Four-dimensional anisotropic mesh adaptation for spacetime numerical simulations*. PhD thesis, Massachusetts Institute of Technology, 2019.

[23] Corentin Carton de Wiart, Laslo T Diosady, Anirban Garai, Nicholas K Burgess, Patrick J Blonigan, Dirk Ekelschot, and Scott M Murman. Design of a modular monolithic implicit solver for multi-physics applications. In *2018 AIAA Aerospace Sciences Meeting*, page 1400, 2018.

[24] IB Celik, ZN Cehreli, and I Yavuz. Index of resolution quality for large eddy simulations. 2005.

[25] Marco Ceze and Krzysztof J Fidkowski. Drag prediction using adaptive discontinuous finite elements. *Journal of Aircraft*, 51(4):1284–1294, 2014.

[26] Guodong Chen and Krzysztof J Fidkowski. Output-based adaptive aerodynamic simulations using convolutional neural networks. *Computers & Fluids*, 223:104947, 2021.

[27] Michel Crouzeix. *Sur l'approximation des équations différentielles opérationnelles linéaires par des méthodes de Runge-Kutta*. PhD thesis, Université de Paris VI Thèse, 1975.

[28] C Carton De Wiart, Koen Hillewaert, Laurent Bricteux, and Grégoire Winckelmans. Implicit LES of free and wall-bounded turbulent flows based on the discontinuous Galerkin/symmetric interior penalty method. *International Journal for Numerical Methods in Fluids*, 78(6):335–354, 2015.

[29] Zhiwen Deng, Chuangxin He, Yingzheng Liu, and Kyung Chun Kim. Super-resolution reconstruction of turbulent velocity fields using a generative adversarial network-based artificial intelligence framework. *Physics of Fluids*, 31(12), 2019.

[30] Ph Devloo. A fully automatic hp-adaptivity. *Journal of Scientific Computing*, 17:117–142, 2002.

[31] Laslo T Diosady and Scott M Murman. Design of a variational multiscale method for high reynolds number compressible flows. In *21st AIAA Computational Fluid Dynamics Conference*, page 2870, 2013.

[32] Laslo T Diosady and Scott M Murman. DNS of flows over periodic hills using a discontinuous Galerkin spectral-element method. In *44th AIAA Fluid Dynamics Conference*, page 2784, 2014.

[33] Laslo T Diosady and Scott M Murman. Higher-order methods for compressible turbulent flows using entropy variables. In *53rd AIAA Aerospace Sciences Meeting*, page 0294, 2015.

[34] Laslo T Diosady and Scott M Murman. Tensor-product preconditioners for higher-order space–time discontinuous Galerkin methods. *Journal of Computational Physics*, 330:296–318, 2017.

[35] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[36] Jim Douglas and Todd Dupont. Interior penalty procedures for elliptic and parabolic Galerkin methods. In *Computing Methods in Applied Sciences: Second International Symposium December 15–19, 1975*, pages 207–216. Springer, 2008.

[37] Gregory L Eyink, Tom WN Haine, and Daniel J Lea. Ruelle's linear response formula, ensemble adjoint schemes and Lévy flights. *Nonlinearity*, 17(5):1867, 2004.

[38] Yuchen Fan, Honghui Shi, Jiahui Yu, Ding Liu, Wei Han, Haichao Yu, Zhangyang Wang, Xinchao Wang, and Thomas S Huang. Balanced two-stage residual networks for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 161–168, 2017.

[39] Sina Farsiu, Dirk Robinson, Michael Elad, and Peyman Milanfar. Advances and challenges in super-resolution. *International Journal of Imaging Systems and Technology*, 14(2):47–57, 2004.

[40] Krzysztof J Fidkowski. Output-based error estimation and mesh adaptation for steady and unsteady flow problems. *38thAdvanced CFD Lectures Series*, pages 14–16, 2015.

[41] Krzysztof J Fidkowski. Output-based space–time mesh optimization for unsteady flows using continuous-in-time adjoints. *Journal of Computational Physics*, 341:258–277,

2017.

[42] Krzysztof J Fidkowski. Output-based error estimation and mesh adaptation for unsteady turbulent flow simulations. *Computer Methods in Applied Mechanics and Engineering*, 399:115322, 2022.

[43] Krzysztof J Fidkowski, Marco A Ceze, and Philip L Roe. Entropy-based drag-error estimation and mesh adaptation in two dimensions. *Journal of aircraft*, 49(5):1485–1496, 2012.

[44] Krzysztof J Fidkowski and Guodong Chen. Metric-based, goal-oriented mesh adaptation using machine learning. *Journal of Computational Physics*, 426:109957, 2021.

[45] Krzysztof J Fidkowski and David L Darmofal. Review of output-based error estimation and mesh adaptation in computational fluid dynamics. *AIAA journal*, 49(4):673–694, 2011.

[46] Krzysztof J Fidkowski and Yuxing Luo. Output-based space–time mesh adaptation for the compressible Navier–Stokes equations. *Journal of Computational Physics*, 230(14):5753–5773, 2011.

[47] Krzysztof J Fidkowski and Philip L Roe. An entropy adjoint approach to mesh refinement. *SIAM Journal on Scientific Computing*, 32(3):1261–1287, 2010.

[48] Corbin Foucart, Aaron Charous, and Pierre FJ Lermusiaux. Deep reinforcement learning for adaptive mesh refinement. *Journal of Computational Physics*, 491:112381, 2023.

[49] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Super-resolution reconstruction of turbulent flows with machine learning. *Journal of Fluid Mechanics*, 870:106–120, 2019.

[50] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows. *Journal of Fluid Mechanics*, 909:A9, 2021.

[51] Haiyang Gao and ZJ Wang. A residual-based procedure for hp-adaptation on 2-d hybrid meshes. In *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 492, 2011.

[52] Anirban Garai, Laslo Diosady, Scott M Murman, and Nateri K Madavan. Development of a Perfectly Matched Layer Technique for a Discontinuous-Galerkin Spectral-Element Method. In *54th AIAA Aerospace Sciences Meeting*, page 1338, 2016.

[53] Anirban Garai, Laslo T Diosady, Scott M Murman, and Nateri K Madavan. Scale-resolving simulations of bypass transition in a high-pressure turbine cascade using a spectral element discontinuous Galerkin method. *Journal of Turbomachinery*, 140(3):031004, 2018.

[54] Anirban Garai, Scott M Murman, and Nateri K Madavan. Scale-Resolving Simulations of a Fundamental Trailing-Edge Cooling Slot Using a Discontinuous-Galerkin Spectral-Element Method. In *Turbo Expo: Power for Land, Sea, and Air*, volume 58578, page V02CT41A023. American Society of Mechanical Engineers, 2019.

[55] Nicholas J Georgiadis, Donald P Rizzetta, and Christer Fureby. Large-eddy simulation: current capabilities, recommended practices, and future research. *AIAA journal*, 48(8):1772–1784, 2010.

[56] Emmanuil H Georgoulis and Paul Houston. Discontinuous Galerkin methods for the biharmonic problem. *IMA journal of numerical analysis*, 29(3):573–594, 2009.

[57] Emmanuil H Georgoulis, Paul Houston, and Juha Virtanen. An a posteriori error indicator for discontinuous Galerkin approximations of fourth-order elliptic problems. *IMA journal of numerical analysis*, 31(1):281–298, 2011.

[58] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[59] Fernando F Grinstein, Len G Margolin, and William J Rider. *Implicit large eddy simulation*, volume 10. Cambridge university press Cambridge, 2007.

[60] Wagdi G Habashi, Julien Dompierre, Yves Bourgault, Djaffar Ait-Ali-Yahia, Michel Fortin, and Marie-Gabrielle Vallet. Anisotropic mesh adaptation: Towards user-independent, mesh-independent and solver-independent CFD. Part I: General principles. *International Journal for Numerical Methods in Fluids*, 32(6):725–744, 2000.

[61] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[62] JR Herring, D Schertzer, M Lesieur, GR Newman, JP Chollet, and M Larcheveque. A comparative assessment of spectral closures as applied to passive scalar diffusion. *Journal of Fluid Mechanics*, 124:411–437, 1982.

[63] Joel Ho and Alastair West. Field Inversion and Machine Learning for turbulence modelling applied to three-dimensional separated flows. In *AIAA aviation 2021 forum*, page 2903, 2021.

[64] Johan Hoffman. Computation of mean drag for bluff body problems using adaptive dns/les. *SIAM Journal on Scientific Computing*, 27(1):184–207, 2005.

[65] Jonathan R Holland, James D Baeder, and Karthik Duraisamy. Towards integrated field inversion and machine learning with embedded neural networks for RANS modeling. In *AIAA Scitech 2019 forum*, page 1884, 2019.

[66] Thomas JR Hughes, Leopaldo P Franca, and Michel Mallet. A new finite element formulation for computational fluid dynamics: I. Symmetric forms of the compressible Euler and Navier-Stokes equations and the second law of thermodynamics. *Computer methods in applied mechanics and engineering*, 54(2):223–234, 1986.

[67] James M Hyman and Basil Nicolaenko. The Kuramoto-Sivashinsky equation: a bridge between PDE's and dynamical systems. *Physica D: Nonlinear Phenomena*, 18(1-3):113–126, 1986.

[68] Jeremy Ims and Zhi J Wang. A comparison of three error indicators for adaptive high-order large eddy simulation. *Journal of Computational Physics*, 490:112312, 2023.

[69] Jianbo Jiao, Wei-Chih Tu, Shengfeng He, and Rynson WH Lau. Formresnet: Formatted residual learning for image restoration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 38–46, 2017.

[70] Hans Johansen and Phillip Colella. A Cartesian grid embedded boundary method for Poisson's equation on irregular domains. *Journal of Computational Physics*, 147(1):60–85, 1998.

[71] Steven Kast, Krzysztof Fidkowski, and Philip Roe. An unsteady entropy adjoint approach for adaptive solution of the shallow-water equations. In *20th AIAA Computational Fluid Dynamics Conference*, page 3694, 2011.

[72] Christopher A Kennedy and Mark H Carpenter. Diagonally implicit Runge-Kutta methods for ordinary differential equations. A review. Technical report, 2016.

[73] John Kim, Parviz Moin, and Robert Moser. Turbulence statistics in fully developed channel flow at low Reynolds number. *Journal of fluid mechanics*, 177:133–166, 1987.

[74] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[75] Dana A Knoll and David E Keyes. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004.

[76] Lilia Krivodonova, Jianguo Xin, J-F Remacle, Nicolas Chevaugeon, and Joseph E Flaherty. Shock detection and limiting with discontinuous Galerkin methods for hyperbolic conservation laws. *Applied Numerical Mathematics*, 48(3-4):323–338, 2004.

[77] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[78] Daniel J Lea, Myles R Allen, and Thomas WN Haine. Sensitivity analysis of the climate of a chaotic system. *Tellus A: Dynamic Meteorology and Oceanography*, 52(5):523–532, 2000.

[79] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[80] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[81] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[82] Marcel Lesieur. Turbulence in fluids: stochastic and numerical modeling. *NASA STI/Recon Technical Report A*, 91:24106, 1990.

[83] Marcel Lesieur and Olivier Metais. New trends in large-eddy simulations of turbulence. *Annual review of fluid mechanics*, 28(1):45–82, 1996.

[84] Bo Liu, Jiupeng Tang, Haibo Huang, and Xi-Yun Lu. Deep learning methods for super-resolution reconstruction of turbulent flows. *Physics of Fluids*, 32(2), 2020.

[85] Miles J McGruder and Krzysztof Fidkowski. Incremental Super-Resolution Reconstruction for Turbulent Flow on High-Order Discontinuous Finite Elements. In *AIAA SCITECH 2024 Forum*, page 1983, 2024.

[86] Miles J McGruder, Aniruddhe Pradhan, and Krzysztof Fidkowski. A Neural-Network Based Adaptive Discontinuous Galerkin Method for Turbulent Flow Simulations. In *AIAA SCITECH 2023 Forum*, page 1802, 2023.

[87] CP Mellen, J Fröhlich, and W Rodi. Large eddy simulation of the flow over periodic hills. In *16th IMACS world congress*, pages 21–25. Lausanne, Switzerland, 2000.

[88] Parviz Moin and John Kim. Numerical investigation of turbulent channel flow. *Journal of fluid mechanics*, 118:341–377, 1982.

[89] Robert D Moser, John Kim, and Nagi N Mansour. Direct numerical simulation of turbulent channel flow up to Re τ= 590. *Physics of fluids*, 11(4):943–945, 1999.

[90] Rodrigo Costa Moura, Gianmarco Mengaldo, Joaquim Peiró, and Spencer J Sherwin. On the eddy-resolving capability of high-order discontinuous Galerkin approaches to implicit LES/under-resolved DNS of Euler turbulence. *Journal of Computational Physics*, 330:615–623, 2017.

[91] Rodrigo Costa Moura, Spencer J Sherwin, and Joaquim Peiró. Linear dispersion–diffusion analysis and its application to under-resolved turbulence simulations using discontinuous Galerkin spectral/hp methods. *Journal of Computational Physics*, 298:695–710, 2015.

[92] Fabio Naddei, Marta de la Llave Plata, Vincent Couaillier, and Frédéric Coquel. A comparison of refinement indicators for p-adaptive simulations of steady and unsteady flows using discontinuous Galerkin methods. *Journal of Computational Physics*, 376:508–533, 2019.

[93] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[94] Angxiu Ni and Qiqi Wang. Sensitivity analysis on chaotic dynamical systems by Non-Intrusive Least Squares Shadowing (NILSS). *Journal of Computational Physics*, 347:56–77, 2017.

[95] Eric J Parish and Karthik Duraisamy. A paradigm for data-driven predictive modeling using field inversion and machine learning. *Journal of computational physics*, 305:758–774, 2016.

[96] Sung Cheol Park, Min Kyu Park, and Moon Gi Kang. Super-resolution image reconstruction: a technical overview. *IEEE signal processing magazine*, 20(3):21–36, 2003.

[97] Jaime Peraire, Morgan Vahdati, Ken Morgan, and Olgierd C Zienkiewicz. Adaptive remeshing for compressible flow computations. *Journal of computational physics*, 72(2):449–466, 1987.

[98] Per-Olof Persson and Jaime Peraire. Sub-cell shock capturing for discontinuous Galerkin methods. In *44th AIAA aerospace sciences meeting and exhibit*, page 112, 2006.

[99] Aniruddhe Pradhan and Karthik Duraisamy. Variational multiscale super-resolution: A data-driven approach for reconstruction and predictive modeling of unresolved physics. *International Journal for Numerical Methods in Engineering*, 124(19):4339–4370, 2023.

[100] Rolf Rannacher. Adaptive Galerkin finite element methods for partial differential equations. *Journal of Computational and Applied Mathematics*, 128(1-2):205–233, 2001.

[101] Béatrice Rivière, Mary F Wheeler, and Vivette Girault. A priori error estimates for finite element methods based on discontinuous approximation spaces for elliptic prob-

lems. *SIAM Journal on Numerical Analysis*, 39(3):902–931, 2001.

[102] Robert S Rogallo and Parviz Moin. Numerical simulation of turbulent flows. *Annual review of fluid mechanics*, 16(1):99–137, 1984.

[103] Youcef Saad and Martin H Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[104] Yukiko S Shimizu and Krzysztof Fidkowski. Output error estimation for chaotic flows. In *46th AIAA Fluid Dynamics Conference*, page 3806, 2016.

[105] Yukiko S Shimizu and Krzysztof Fidkowski. Output-based error estimation for chaotic flows using reduced-order modeling. In *2018 AIAA Aerospace Sciences Meeting*, page 0826, 2018.

[106] SUN Shuyu and Mary F Wheeler. Mesh adaptation strategies for discontinuous Galerkin methods applied to reactive transport problems. 2004.

[107] Anand Pratap Singh, Shivaji Medida, and Karthik Duraisamy. Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils. *AIAA journal*, 55(7):2215–2227, 2017.

[108] Joseph Smagorinsky. General circulation experiments with the primitive equations: I. The basic experiment. *Monthly weather review*, 91(3):99–164, 1963.

[109] Endre Süli and Igor Mozolevski. hp-version interior penalty DGFEMs for the biharmonic equation. *Computer methods in applied mechanics and engineering*, 196(13-16):1851–1863, 2007.

[110] Lionel Temmerman and Michael A Leschziner. Large eddy simulation of separated flow in a streamwise periodic channel constriction. In *Second Symposium on Turbulence and Shear Flow Phenomena*. Begel House Inc., 2001.

[111] Kenza Tlales, Kheir-Eddine Otmani, Gerasimos Ntoukas, Gonzalo Rubio, and Esteban Ferrer. Machine learning mesh-adaptation for laminar and turbulent flows: applications to high-order discontinuous Galerkin solvers. *Engineering with Computers*, pages 1–23, 2024.

[112] Siavash Toosi and Johan Larsson. Anisotropic grid-adaptation in large eddy simulations. *Computers & Fluids*, 156:146–161, 2017.

[113] Siavash Toosi and Johan Larsson. Towards systematic grid selection in les: Identifying the optimal spatial resolution by minimizing the solution sensitivity. *Computers & Fluids*, 201:104488, 2020.

[114] Matteo Tugnoli, Antonella Abbà, Luca Bonaventura, and Marco Restelli. A locally p-adaptive approach for Large Eddy Simulation of compressible flows in a DG framework. *Journal of Computational Physics*, 349:33–58, 2017.

[115] A Uranga, P-O Persson, M Drela, and Jaime Peraire. Implicit large eddy simulation of transition to turbulence at low Reynolds numbers using a discontinuous Galerkin method. *International Journal for Numerical Methods in Engineering*, 87(1-5):232–261, 2011.

[116] David A Venditti and David L Darmofal. Grid adaptation for functional outputs: application to two-dimensional inviscid flows. *Journal of Computational Physics*, 176(1):40–69, 2002.

[117] Qiqi Wang, Rui Hu, and Patrick Blonigan. Least squares shadowing sensitivity analysis of chaotic limit cycle oscillations. *Journal of Computational Physics*, 267:210–224, 2014.

[118] GARY WARREN, Wit Anderson, JAMES THOMAS, and Sherrie Krist. Grid convergence for adaptive methods. In *10th Computational Fluid Dynamics Conference*, page 1592, 1991.

[119] You Xie, Erik Franz, Mengyu Chu, and Nils Thuerey. tempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow. *ACM Transactions on Graphics (TOG)*, 37(4):1–15, 2018.

[120] Jiayang Xu, Aniruddhe Pradhan, and Karthikeyan Duraisamy. Conditionally parameterized, discretization-aware neural networks for mesh-based modeling of physical systems. *Advances in Neural Information Processing Systems*, 34:1634–1645, 2021.

[121] Jiachen Yang, Tarik Dzanic, Brenden Petersen, Jun Kudo, Ketan Mittal, Vladimir Tomov, Jean-Sylvain Camier, Tuo Zhao, Hongyuan Zha, Tzanio Kolev, et al. Reinforcement learning for adaptive mesh refinement. In *International Conference on Artificial Intelligence and Statistics*, pages 5997–6014. PMLR, 2023.

[122] Wenming Yang, Xuechen Zhang, Yapeng Tian, Wei Wang, Jing-Hao Xue, and Qingmin Liao. Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia*, 21(12):3106–3121, 2019.

[123] Masayuki Yano and David L Darmofal. An optimization-based framework for anisotropic simplex mesh adaptation. *Journal of Computational Physics*, 231(22):7626–7649, 2012.

[124] Masayuki Yano et al. *An optimization framework for adaptive higher-order discretizations of partial differential equations on anisotropic simplex meshes*. PhD thesis, Massachusetts Institute of Technology, 2012.

[125] Anil Yildirim, Gaetan KW Kenway, Charles A Mader, and Joaquim RRA Martins. A Jacobian-free approximate Newton–Krylov startup strategy for RANS simulations. *Journal of Computational Physics*, 397:108741, 2019.